머 리 말

1977년 4월 코모도어사가 PET라는 키보드, 디스플레이 및 보조 기억 장치인 카세트 테이프를 붙인 마이크로 컴퓨터를 발표, 그와 동시에 애플사에서는 APPLE Ⅱ, 8월에는 댄디 라디오 샤크사의 TRS-80이 발표되었다.

그 후 마이크로 컴퓨터는 하드웨어에서의 반도체 제조기술의 비약적인 진보로 그 생산 단가의 엄청난 절감과 연산속도의 증가, 크기의 놀랄만한 축소로 인해 개인도 퍼스널 컴퓨터(personal computer)라는 마이크로 컴퓨터를 소유하게 되었다. 이러한 퍼스널 컴퓨터는 소유한 사람의 독자적인 사무실로서의 역할을 충족시켜 주게 되었고 나아가 인간과 기기와의 우의를 증진시키는 역할도 하게 되었다. 따라서 마이크로 컴퓨터는 우리 생활에 점차 뿌리를 내리고 있고 그에 대한 의존도 또한 날로 증대하고 있는 실정이다.

컴퓨터를 보다 잘 이용하기 위해서는 그 컴퓨터의 구성이나 원리 및 사용법, 특히 프로그래밍에 대해 숙달되어야 한다. 퍼스널 컴퓨터의 사용법은 비교적 간단하여 프로그래밍을 제외한 조작법은 몇 시간 정도로 충분히 익힐 수 있다.

이 책에서는 연산속도가 빠르고 미려한그래픽등으로전세계적인 인기를 모으고 있는 APPLE Ⅱ 컴퓨터에서 사용하는 Applesoft에 따랐으며, 일반인도 누구나 이해 할 수 있도록 메뉴얼 형식의 해설과 사용법을 다루었다. 특히 BASIC의 기본적인 사용법에서부터 차근차근 수준을 높일 수 있도록 하였다. 또 일반적인 BASIC과는 다른 특징으로서 컬러 그래픽에 대한 상세한 용어 해설과 함께 이를 응용할 수 있게 하였다.

그리고 많은 예제 프로그램과 연습문제를 통해 컴퓨터가 없어도 전체를 이해할 수 있도록 하였다. 따라서 APPLE 기종이 아닌 다른 기종을 갖고 있는 독자도 자신이 가지고 있는 컴퓨터의 BASIC 메뉴얼을 참고하면 쉽게 응용될 수 있을 것으로 믿는다.

끝으로 본서의 집필에 있어 국내외의 여러 저서와 문헌을 참고하였기에 이들의 저자들에게 충심으로 사의를 표하는 바이다.

이 책의 내용에서 자료의 미흡과 저자 자신의 경험 부족으로 인하여 미미한 점이 많으리라 생각되나 다만 본서가 BASIC 프로그래밍을 공부하는데 여러분에게 조금이라도 도움이 된다면 그보다 큰 기쁨이 없겠다.

그리고 이 책이 나오기까지 적극적으로 협조하여 주신 韓國理工學社 劉光鍾 사장님과 편집부 여러분에게 깊은 감사를 드리는 바이다.

1984. 3.

著者 識

참 고 문 헌

1. Michael P. Barnett & Graham K. Barnett "Apple Ⅱ Graphic" Little, Brown and Company, Toronto, Canada, 1983
2. Roy E. Myers "Microcomputer Graphics" with Apple Ⅱ examples, Addison Wesley Comp. In., 1982
3. B. J. Korites "Graphic Software for Microcomputers" Kern Publications, Duxbury, 1981
4. W. M. Newman & R. F. Sproull "Principles of Interactive Computer Graphics" Mc Graw-Hill, New York, 1979
5. Scot Kamins & Mitchell Waite "Apple Backpack" humanized programming in BASIC, BYTE BOOKS, McGraw-Hill, Peterborough, 1982
6. Roger Wagner "Assembly Lines" for Apple Ⅱ, Softalk Publishing, North Hollywood, 1982
7. Rodnay Zaks "Programming the 6502", SYBEX Inc., 1980
8. "Apple Ⅱ Reference Manual" Apple Computer Inc., 1979
9. "Apple Ⅱ BASIC Programming Reference Manual" Apple Computer Inc., 1979
10. "The Applesoft Tutorial" Apple Computer Inc., 1979
11. "The DOS Manual" Apple Computer Inc., 1981
12. Seamus Dunn & Valerie Morgan "The Apple Personal Computer for Beginner" Prentice/Hall International
13. "MZ-80 series BASIC 解說" SHARP 株式會社刊, 1980
14. Byron S. Gottfried "Programming with BASIC" Shaum's outline series, McGraw-Hill, 1975
15. 鄭天錫 외 1 명 "Applesoft BASIC Programming" 尙學堂, 1982. 3.
16. "Applesoft BASIC 프로그래밍" 홍능과학 출판사 편집부, 1983. 8.
17. 尹正模 외 2 명 "BASIC 프로그래밍" 韓國理工學社, 1983. 6.
18. 黃熙隆 외 1 명 "APPLE Ⅱ BASIC 프로그래밍 500題" 教學社, 1983. 6.
19. 이병태 외 1 명 "APPLE 입문" 機電研究社, 1983
20. Apple Ⅱ 동호인회 편저 "Apple 操縱法入門" 清明出版社, 1983
21. 月刊 "마이크로 소프트웨어" 정보시대 刊, 1983. 11~1984. 2

차　　례

1. BASIC PROGRAMMING의 기본 ▷

◁ 2. 그래픽의 특성 및 활용

◁ 3. 특수기능

4. 디스켓의 사용방법

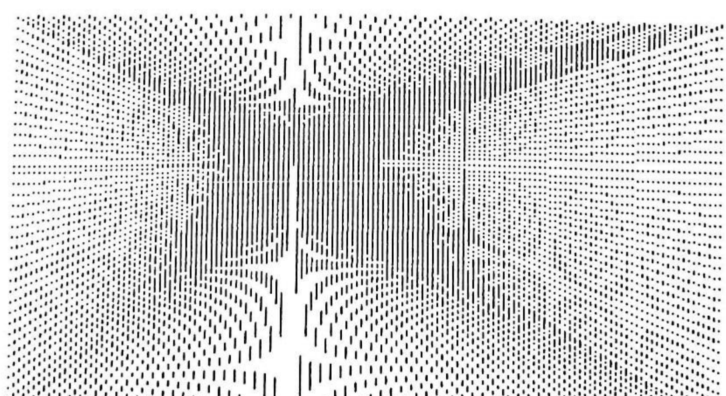
5. 응용 및 게임(Game)

부 록

찾아보기

apple

BASIC PROGRAMMING 의 기본



1 - 1 키보드의 사용법

1-1-1 APPLE Ⅱ의 시동

애플 컴퓨터 뒤에 있는 전원(power) 스위치를 올려보자. "삑"하는 경쾌한 음과 함께 화면에는 「APPLE Ⅱ」라는 「] 」모양의 **프롬프트**(prompt)와 깜박이는 정사각형인 **커서**(cursor)*가 나타날 것이다.

*註 CRT상의 입출력 위치를 사용자에게 나타나게 하는 신호

```
              APPLE ][

  ]■
```

자, 이제 컴퓨터는 당신의 명령을 따를 준비가 다 되었다는 신호를 보내왔다. 무엇인가 프로그램을 수행하고 있지 않은 동안에 커서의 좌측에는 또 하나의 다른 문자가 항상 표시되어 있다. 이것을 프롬프트라고 하며 어떤 언어방식을 사용하고 있느냐와 프로그램 문장의 시작부분을 표시한다.

표1-1 사용될 언어와 표시되는 프롬프트 기호

프롬프트	사용될 언어
*	Monitor mode
]	APPLESOFT BASIC
〉	Integer BASIC
!	Mini Assembly

여기서 *는 모니터 프롬프트로서 어셈블리어를 사용할 수 있고 모니터 프로그램과의 직접통신을 할 수 있다.

이 책에서는 APPLESOFT BASIC을 쓰기 때문에 프롬프트로 「] 」가 나타난다. 이 상태에서 모니터 방식으로 들어가려면

```
]CALL-151

*
```

반대로 모니터방식에서 BASIC방식으로 되돌아갈 때는 CTRL-B 나 CTRL-C 로서 다시 프롬프트로 「] 」문자가 나타난다.

그림1-1과 같은 애플의 키보드*의 영문자 배열은 표준타자기와 똑같고 숫자는 맨 윗줄에 1에서 0까지 일렬로 배열되어 있다. 이것들은 그냥 키를 누르면 해당 문자가 화면에 찍혀져 나타날 것이다.

*註 모든 데이터 또는 명령어를 입력시킬 수 있는 기본장치로서 한글, 영문 숫자의 문자판을 말한다.

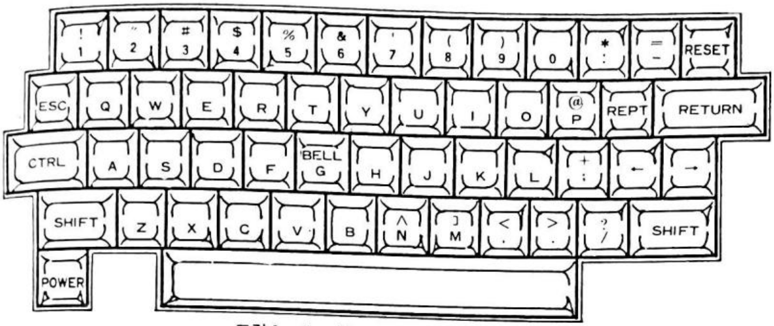


그림 1-1 APPLE Ⅱ PLUS 키보드 배열

1-1-2 키보드의 사용법

RESET키

[RESET] 키는 매우 독특한 키인데 이것을 누르면 무엇을 하고 있다가도 컴퓨터의 동작이 중단되고 콘트롤이 키보드로 넘어간다. 즉 키보드를 사용하여 입력되는 새로운 명령을 받아들일 준비를 갖추게 되는 것이다. 키보드와 같이 작동하고 있던 라인프린터도 이 리셋키에 의해서 중지된다.

RETURN키

[RETURN] 키는 리셋키 바로 아래에 있는데 그 때까지 당신이 타이핑해 놓은 글자들이 기억 장치내로 들어가면서 한 문장이 끝났다는 것을 컴퓨터에게 알려주는 구실을 하게 된다. 애플 소프트의 경우, 잘못 타이핑된 문장이 있으면 그 문장을 수용하기 직전에

```
?SYNTAX ERROR IN XXX
```

라는 메시지가 벨소리와 함께 나타난다. 이 때 ×××의 자리에 해당되는 행번호가 나타나게 되는 것이다.

문장을 원하는대로 타이프하고 반드시 이 [RETURN] 키를 눌러야 한다. 따라서 키보드에서 입력된 문자는 이 [RETURN] 키를 누르기까지는 아무 의미도 없다. 만약 입력된 것이 **직접명령**(direct command)이면 [RETURN] 키를 누르는 것으로 실행이 시작하고 행번호가 붙은 프로그램의 문장으로서 입력되면 프로그램의 일부로서 메모리내에 스토어(store) 된다. 언제고 [RETURN] 키를 누르는 것으로 입력의 종료가 판단되고 처리로 이동한다.

SHIFT키

[SHIFT] 키는 일반타자기에서는 대문자와 소문자간의 변환을 시키기 위한 키인데 키의 윗부분에 표시된 기호를 찍는데 사용된다.

예를 들면 [#/3] 키를 누르면 3 이 찍히지만 [SHIFT] 를 누른채 [#/3] 키를 누르면 # 가 찍히게 된다. 어떤 키들은 [SHIFT] 를 누르고 하나, 안누르고 하나 변화가 없는 것들도 있다.

예외적으로 [SHIFT] - [G] 를 누르면 BELL이란 글자가 찍히지 않고 그냥 G가 찍히는데 이는 [CTRL] 키 부분에 가서 다시 설명이 될 것이고 [SHIFT] - [M] 은]

가 찍히게 된다.

항상 SHIFT 를 누른채 다른 키를 눌러야 키 윗쪽 부분의 부호를 찍는 동작이 일어날 것이다.

CTRL키

이것도 SHIFT 키와 마찬가지로 CTRL 키를 누른채 다른 키를 눌러야 원하는 동작이 일어난다.

몇가지 예를 보자.

CTRL - B : 어셈블리어 사용형태에서 BASIC으로 돌아가고자 할때

CTRL - X : 어떤 문장을 타이핑하다가 그것을 전부 무시해 버리고 싶을 때 사용되며, 그 결과로 CTRL - X 를 누른 직후 커서가 있던 자리에\ 가 찍히며 커서는 다시 다음줄 맨 앞으로 이동된다. 화면에 글자가 남아 있어도 컴퓨터는 그것을 전혀 없는것으로 무시해 버린다.

CTRL - C :BASIC프로그램이 수행되고 있는 동안 이것을 누르면 프로그램이 그 자리에서 수행 정지되며 벨소리와 함께

```
BREAK IN XXX
```

라는 메시지가 찍힌다. 다시 그 다음부터 수행시키려면 CONT를 타이프하고 RETURN 키를 누르면 수행이 계속된다.

CTRL -S : 프로그램을 LIST할때는 LIST를 타이핑하고 RETURN키를 누르면 되는데, 이때 화면상으로 프로그램이 너무빨리 찍혀져 지나가므로 볼 수가 없다. CTRL - S를 누르면 화면상에 정지한다. 화면상에 찍히는 출력도 마찬가지이다. 계속보려면 아무키나 누르면 된다.

CTRL -G : 이것은 BELL이란 글자가 찍히는게 아니고 스피커에서 벨소리는 아니지만 "삑"하는 소리를 낸다. 프로그램중에 이것을 삽입하므로써 출력시에 소리를 들을 수도있다.

ESC키

이것은 SHIFT 나 CTRL 키와는 달리 ESC 를 누른채 다른 키를 누르지는 않는다. 즉 ESC 키를 눌렀다 놓은 다음 다른 키를 누르는데 이 동작을 **이스케이프 시퀀스** (escape sequence)라 부른다.

예로써 화면을 깨끗이 지우고 커서를 왼쪽위로 옮기려면, 이 상태를 컴퓨터 용어로 HOME이라 부르는데 이때 ESC - @를 누르면 된다.

ESC　SHIFT
　　　@/P

먼저 ESC 를 눌렀다 놓는다. 다음 @를 누르려면 SHIFT P 를 눌러야 한다.

ESC A : 커서가 오른쪽으로 1문자이동하고 키입력 대기로 된다.

ESC B : 커서가 왼쪽으로 1문자이동하고 키입력대기로 된다.

ESC C : 커서가 아래쪽으로 1문자이동하고 키입력대기로 된다.

ESC D : 커서가 위로 1문자이동하고 키입력대기로 된다.
ESC E : 커서가 있는 위치에서 그행의 끝까지를 소거한다.
ESC F : 커서가 있는 위치에서 화면의 최종까지를 소거한다.
ESC @ : 화면전체를 소거한다.
ESC I : 커서를 계속하여 위쪽으로 이동한다.
ESC J : 커서를 계속하여 왼쪽으로 이동한다.
ESC K : 커서를 계속하여 오른쪽으로 이동한다.
ESC M : 커서를 계속하여 아래쪽으로 이동한다. *

* 註 auto, start, ROM 에만 있는 기능으로 I J K M REPT CTRL SHIFT 이외의 키를 누를 때까지 이 방식에서 빠져 나오지 않는다.

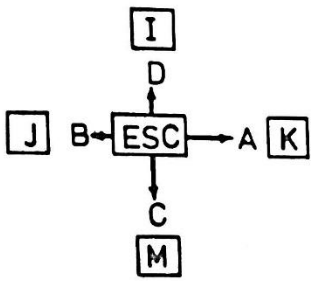


그림 1-2 ESC에 의한 커서의 이동

←→ 키

이 키들은 타이핑을 잘못해 넣은 뒤 수정할 때 주로사용된다. ←를 누르면 커서가 왼쪽으로 한 칸 이동하므로 원하는 자리에 가서 다시치면 수정된다.

→키는 커서가 오른쪽으로 한 칸 이동하되 커서가 지나온 부분의 문자를 그대로 복사하는 것처럼 기억한다.

스페이스바를 눌러도 한 칸 우측으로 가지만 지나온 자리가 빈칸으로 남고 내용이 지워져 버린다.

RETURN 키를 누르면 커서가 있는 자리부터 우측 이후의 모든 문자가 없어지므로 수정한 후 적당한 위치로 →를 사용하여 되돌아 온 후 RETURN 키를 눌러야 한다.

REPT키

이것은 똑같은 키를 여러번 눌러야할 때 그 번거로움을 덜어준다. 원하는 키를 누르고 REPT키를 누르면 누르고 있는동안 같은키를 여러번 반복하여 타이핑하는 효과를 가져온다. →나 ←와 함께 사용하면 아주 유용하다.

1-1-3 프로그램의 기록

LOAD

프롬프트와 깜박거리는 커서는 애플소프트의 시작임을 나타낸다. 이제 당신은 테이프 레코더에 볼륨 컨트롤을 세트할 준비가 되었다.

테이프 레코더를 돌릴때 보통은 당신에게 들리는 소리를 낸다는 목적을 가지고 있다. 그것이 너무 작으면 말이나 음악이 들리지 않는다. 너무 크면 귀찮다. 이

런 경우 에러 메시지를 내게 된다.

바른 볼륨의 세팅을 발견하기 위해 대수롭지 않은 시행착오를 반복하지 않으면 안된다. 애플소프트의 테이프는 컴퓨터에 볼륨을 작게해서 넣어서 정보가 무사히 들어 갔는지 어떤지 상황을 본다. 만약 작용하지 않으면 이번에는 볼륨을 조금 더 올려 다시 한번 테이프를 돌려본다. 볼륨이 맞을 경우 컴퓨터는 삑하는 소리와 함께 정보가 들어가고 있음을 알린다.

하여간 처음에 스크린을 클리어 하자.

ESC SHIFT @P

볼륨 컨트롤의 위치를 맞추기 위해

1. 테이프를 맨처음까지 되돌려 감는다.
2. 다음과 같이 타이프한다.
   L O A D
3. 레코더의 PLAY를 눌러 테이프를 돌린다.
4. RETURN 키를 누른다.

이것을 하면 커서는 사라지고 무엇인가 일어날때까지 15초가 걸린다.

그리고 나서 다음의 어느 것이 된다.

a.
```
?SYNTAX ERROR
]
```

의 메세지가 나온다.

b. 전혀 아무것도 일어나지 않는다.

c.
```
ERR
```

또는

```
ERRERR
```

(삑하는 소리가 있든지 또 없이) 나타난다.

d 컴퓨터가 삑 소리를 내고 아무것도 나타나지 않는다.

여기서 a의 경우 볼륨 컨트롤을 바꾸지 말고 테이프를 되돌려 감고 다시 시작한다.

b와c의 경우 15초 이후까지 기다렸다가 프롬프트나 커서가 나와있지 않고 키보드를 눌러도 아무 반응이 없다면 RESET 키를 눌러 볼륨을 조금 높이해서 다시 시도한다. 아주 드물게 LOAD의 명령이 바르게 작용하지 않았는지도 모른다. 그러면 테이프가 LOAD되는 것을 기다리지 않고 커서가 즉시 화면에 나타날 것이다.

그렇게 되면 컴퓨터의 스위치를 끄고 다시 한번 컴퓨터의 전원 스위치를 ON으로 다시 시작하라.

d의 경우 정보가 완전히 들어가게 되면 한번 더 삑소리를 내면서 프롬프트와 커서가 나타나는데 이 경우는 완전히 LOAD된 것이다.

5. 테이프 레코더를 멈추고 테이프를 되돌려 감는다. 정보가 컴퓨터에 들어갔으므로 테이프 레코드는 다음까지 사용할수없다.

6. R U N

을 타이프하고 RETURN 키를 누름으로써 프로그램은 실행된다.

그러나 d의 경우가 되어서 조금 시간이 지난후 c처럼 ERR 메세지가 나타나면 테이프를 서게하고 되돌려감아 다시 시작해야 한다.

만일 완전히 LOAD되었다면 앞으로 계속 테이프를 LOAD할 때 그 세팅을 이용할 수 있도록 레코더의 볼륨컨트롤의 위치에 표시를 해두는 것이 좋을 것이다.

당신의 컴퓨터에 애플소프트를 사용해야 하므로 로딩(roading)하는 테이프도 애플소프트로 쓰여 있지 않으면 안된다. 그렇지 않으면 로딩되는 결과로서 모니터 화면에는 기묘한 틀림의 메세지나 기호가 나타날지 모른다. 또 키보드의 컨트롤을 잃을지도 모르고 더 여러가지 일이 일어날지도 모른다. 이것이 일어나면 컴퓨터의 스위치를 끄고 그리고 모든것이 원래로·되돌아 가도록 또 스위치를 넣어라.

SAVE

그리고 당신이 뒤에 사용하고 싶은 프로그램을 테이프레코더에 넣고 처음까지 되돌려 감는다. 이렇게 하므로 프로그램을 찾기 쉽기 때문이다. 물론 카운팅이 붙어있는 레코더는 카운팅되는 숫자를 기억하면 더욱 좋은 방법이 될 것이다.

자, 프로그램을 SAVE 해 보자.

1. 다음과 같이 타이프한다.

   S A V E

2. 테이프레코더의 REC PLAY 단자를 동시에 누른다.
3. 컴퓨터의 RETURN 키를 누른다.
4. SAVE가 시작되면 삑소리가 난다.
5. SAVE가 끝나면 역시 삑소리를 내면서 프롬프트와 커서가 나타난다.



LOAD, SAVE

6. 레코더의 [STOP] 버튼을 누른다.

7. 테이프를 처음까지 되돌려 감는다.

이것으로 당신은 프로그래밍으로 되돌아 갈 수 있다. 컴퓨터 내의 프로그램은 SAVE해도 아무 영향도 받지 않는다.

1-1-4 프로그램의 수정

```
* PROGRAM 1.1
 10 A = 3:B = 7:C = 4
 20  PRINT A * B + C
```

위에서 행번호 10의 B의 값을 5로 바꾸고자 할 때 제일 밑의 커서를 행번호 10의 7의 위치로 이동시켜야 할 것이다.

[ESC] 키를 누른 다음 [J]를 눌러 커서가 프롬프트(]) 위로 이동시킨다. 다음에 [I]를 계속 눌러 (이때 [REPT] 키도 같이 누른다.) 커서가 위로 올라가도록 한다.

```
10 A = 3:B = 7:C = 4
```

의 위치일때 [ESC] 키를 다시 누른다. 이 때의 [ESC] 키를 누르는 것은 **이스케이프 루틴**(escape routine)에서 빠져나오는 동작이다. 다음 [→]키를 반복해서 눌러 7이 타이프된 곳까지 커서를 오른쪽으로 이동시킨다. 커서가 7의 위치에 있게 되면 [5]를 타이핑한다. 7이 5로 수정되면 다시 [→] 키를 눌러 끝까지 복사시켜 4다음에 커서가 있게 되면 [RETURN]키를 누르므로 수정이 끝난다. 이러한 수정행위를 **스크린 에디팅**(screen editing)이라 부른다.

이런 과정은 프로그램이 길 때는 에러가 발생될 수가 있으므로 단지 수정할 행번호만 꺼집어 내어 스크린 에디팅 시켜도 좋을 것이다.

```
]LIST10        [RETURN]
```

이라 하면

```
10 A = 3:B = 7:C = 4
```

의 문장이 나타날 것이다. 여기서 똑같은 형태로 실행시켜 프로그램을 수정할 수도 있다. 수정한 후에 확인하기 위해서 다음과 같이 LIST 10을 타이핑하고 [RETURN] 키를 누르면 수정을 확인할 수 있다.

```
]LIST10

10 A = 3:B = 5:C = 4
```

1-2 기본 프로그램

PRINT

지금 스크린상에 프롬프트(prompt ; 기호])와 점멸하는 커서(cursor)가 있으면 이것으로 당신은 당신의 컴퓨터를 사용하기 시작할 준비가 된 것이다.

```
]PRINT "APPLE"    RETURN
```

라고 타이프하면 컴퓨터는 다음 행에

```
APPLE
```

이라고 프린트 한다. 이 때 PRINT를 틀리게 타이프하면 에러메세지인

```
]PRIMT "APPLE"

?SYNTAX ERROR
```

가 표시될 것이다. 처음의 쿼트(quote ; 기호 ") 혹은 양쪽 쿼트를 다 타이프하지 않으면 컴퓨터는

```
]PRINT APPLE"
0

]PRINT APPLE
0
```

로 프린트한다. 제로는 슬래시로 알파벳 O와 구별한다.*

RETURN 키 앞의 쿼트는 특별히 타이프 할 필요가 없다. 예를 들면

```
]PRINT "APPLE
APPLE
```

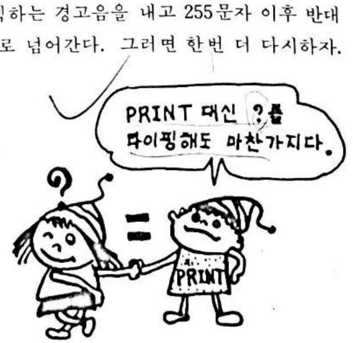
처럼 최후의 쿼트는 있어도 없어도 프린트된다. 그러나 닫는 쪽의 쿼트를 언제나 붙이도록 하는 것이 좋을 것이다. 이 습관은 나중에 중요하게 될 것이다.

```
]PRINT "APPLE"
```

이것은 쿼트로 둘러싸인 문자 모두를 스크린상에 표시하도록 컴퓨터에 말하는 명령이다. PRINT 명령을 사용해서 컴퓨터에 원하는 메세지를 표시 시킬 수 있다. 그러나 240문자를 넘으면 컴퓨터는 삑하는 경고음을 내고 255문자 이후 반대방향 슬래시(\)를 표시하면서 다음 행으로 넘어간다. 그러면 한번 더 다시하자.

```
]PRINT "32"
32
```

*註
이 책에서는 프린트상 수치0은 0, 알파벳O는 O로 구분한다.



PRINT

컴퓨터는 기대대로 다음 행에 32라고 프린트 하는데

```
]PRINT 32
32
```

와 같이 쿼트가 없는 **스테이트먼트** (statement) 라도 똑 같이 수치를 프린트하고 경고음이나 에러 메세지는 표시되지 않는다. 실은 컴퓨터에는 쿼트로 전연 둘러 싸지 않아도 수치를 프린트 시킬 수 있다.

직접 명령식 PRINT문을 한번 해보자.

```
]PRINT 3+5
8
```

다음 행에 8 이라고 숫자가 나온다. 컴퓨터는 5 가지의 수치연산을 할 수 있다.

1. 덧셈은 플러스 기호(+)를 사용한다.

2. 뺄셈은 마이너스 기호(−)를 사용한다.

3. 곱셈은 **애스터리스크** (asterisk ; 기호 *) 를 사용한다. 왜냐하면 보통 곱셈에서는 ×를 사용하나 이것은 알파벳 X와 구별이 어렵고 또 도트(·)는 피어리드나 소숫점과 틀리기 쉽기 때문이다. 따라서 7 곱하기 8 은

```
]PRINT 7*8
56
```

와 같이 애스터리스크(*)를 사용하기로 한다.

4. 나눗셈은 슬래시(slash : 기호 /) 를 사용한다. 35나누기 7 은

```
]PRINT 35/7
5
```

즉 5 라고 답이 실행된다.

5. 지수함수 이용 방식이다. 이것은 같은 수를 몇번이고 곱셈하는데 편리하다.

```
]PRINT 3*3*3*3*3
243
```

이러한 번거로움도

```
]PRINT 3^5     ∧
243
```

이것으로 된다. 위의 ∧기호는 이렇게 타이프 한다.

[SHIFT]

[∧ N]

즉, [SHIFT] 키를 누르면서 [∧ N]를 누른다.

지수에 대해서는 무슨 특별한 것은 없다. 다만 곱셈을 생략했다고 생각하면 된다. 3^5처럼 쓰는 것은 컴퓨터에서는 불가능하기 때문이다.

컴퓨터에서의 수치 표현은

```
]PRINT 79.290
```

이라고 타이프하면 컴퓨터는

```
79.29
```

이라고 답하고 마지막의 0는 프린트 하지 않는다. 즉 최초나 최후의 0는 프린트 되지 않는다. 이것은 소수표시의 정수부가 0이거나 소수부의 마지막이 0이라고 표시하지 않는다는 것이다.

```
]PRINT 985788.6898
985788.69
```

을 시도해 보면 뒤의 2자리가 없다. 마지막 자리는 컴퓨터가 생각한 근사치이다. 이것을 끝수처리라고 한다.

```
]PRINT 788.6987
788.6987
```

이번에는 끝수처리가 생기지 않고 타이프한대로 표시되었다. 이것은 수치가 10자리 이상일 때 끝수처리가 되고 10자리 미만의 수치에서는 생기지 않기 때문이다. 따라서

```
]PRINT 1234567890
1.23456789E+09
```

처럼 10자리 일 때 이런 모양으로 인자된다.

1234567890과 1.23456789E+09는 같은 수치이다. 컴퓨터의 수치는 **과학표기법**으로 표시하기 때문이다.

Applesoft에서는 부동 소숫점 연산을 하고 통상 에러가 나오지 않는 범위는 -1 *10∧38부터 1 *10∧38 ($-1\times10^{38}\sim1\times10^{38}$) 인데 가감산의 결과로서 $1.7*10^{38}$까지의 크기의 수를 허용하는 수도 있다. 또 절대치가 3 *10∧-39 (3×10^{-39})보다 작은 수는 0으로 간주한다.

여기서 참(true)의 정수의 범위는 -32767~32767이다.

수치가 출력될 때에는 다음의 규칙에 의해서 그 포맷이 결정된다.

1. 수치가 음의 경우는 수치의 바로 앞에 붙는다.
2. 절대치가 0~999999999의 범위에 있는 수치는 소숫점이 없이 정수형을 취해 표시된다.
3. 절대치가 0.01~99999999.2의 범위에 있는 수치는 고정 소숫점 출력이 된다.
4. 위의 2.3항에 적합치 않은 수치는 과학표기형으로 표시된다.

과학 표기법은 실수형의 수치를 나타낼 때 사용되며

```
SX.XXXXXXXXESTT
```

라는 포맷을 가지고 있고 기호는 다음과 같은 의미를 갖는다.

S … 부호를 나타낸다(최초의 S는 가수부로 +일 때는 수학상의 관습으로 생략되고 −일때는 표시된다. 2번째는 진수부의 부호인 +, −가 표시된다).

X … 0부터 9까지의 1자리의 정수를 나타낸다(가수부).

E … 여기서 부터 뒤가 1.0의 누승수(진수)인 것을 나타내며 표시에서도 E로 그대로 표시된다.

T … 0부터 9까지의 1자리의 정수를 나타낸다(진수부). 또, 항상 2자리의 출력이 행해진다.

과학 기술 표시에서는 소숫점 앞이 0으로 되는 일이 없으며 따라서 부호 바로 뒤의 소숫점 이상의 수X는 절대로 0으로 되지 않는다.

다음은 여러가지 수치가 어떻게 출력되는지의 예이다.

수	출력포맷
+1	1
−1	1
6523	6523
−23.460	−23.46
45.72E5	4572000
1 * 10∧20	1 E+20
−12.34567896 * 10∧10	−1.23456789E+11
1000000000	1 E+09
999999999	999999999

키보드에서 입력되는 문자는 38자리까지 에러없이 입력이 가능하나 최초의 9자리만이 유효하며 10자리째에서 반올림되게 되어 있다. 즉

```
]PRINT 1.23456787654321
```

로 하면

```
1.23456788
```

이 된다.

연산순서

애플소프트 BASIC이 연산할 때는 우선 순위가 있다.

```
]PRINT 4+8/2
```

와 같은 단순한 계산이라도 어떠한 순서로 계산이 실행되는지 알고 있지 않으면 답이 6인지 8인지를 모른다. 4와 8을 합쳐 12, 이것을 2로 나누면 답은 6. 그러나 만약 8을 2로 나눈 것과 4를 더하면 8. 이 모든것이 답이 될 수 있다. 컴퓨터로 계산하면 답은 8이 된다.

그러면 연산 순서를 알아보자.

1. 애플소프트 BASIC에서는 최초의 괄호를 먼저 처리한다.
2. 음의 값을 나타낼 때 마이너스 기호를 사용하면 마이너스 기호를 먼저 읽는

다.

```
-3+2
```

그러므로 이것은 −1이라는 결과가 된다. 만약 컴퓨터가 덧셈을 먼저 실행하면 −3+2는 −5가 되는 것이다. 그러나 실제로 그렇지 않다.

3. 마이너스 기호를 처리하면 컴퓨터는 다음에 지수연산으로 옮긴다. ^

```
4+3^2
```

이것은 우선 3의 제곱(3 * 3 = 9)을 실행하고 4를 더하므로 결과는 13이 된다. 지수가 많을 경우에는 좌측에서 우측으로 처리된다. 예를 들면

```
2^3^2
```

이것은 2의 3제곱(2 * 2 * 2 = 8)을 실행하고 나중에 8에 대한 제곱을 계산한다. 그러므로 답은 64이다.

4. 지수연산이 끝나면 모든 곱셈과 나눗셈이 좌측에서 우측을 향하여 실행된다. 같은 우선도를 가진 연산끼리는 언제나 좌측에서 우측으로 처리된다. 곱셈(*)과 나눗셈(/)은 같은 우선도이다. *, /

5. 최후에 덧셈과 뺄셈이 좌측에서 우측으로 처리된다. 덧셈(+)과 뺄셈(−)은 같은 우선도를 가진다. +, −

그럼 컴퓨터의 연산순서를 정리해 보자.

```
( )
NOT -
^
* /
+ -
> < >= <= => =< <> ><=
AND
OR
```

만약 4에 2를 더한 것으로 12를 나누려고 한다면

```
12/4+2
```

이것은 12를 4로 나눈 것에 2를 더하는 것이 된다. 그러나 이것은 원하는 것과 다르다. 처음 생각한 대로 표현하려면

```
12/(4+2)
```

라고 쓰면 된다. 괄호는 우선이 된다. 즉 괄호안을 제일 먼저 처리한다. 만약 괄호가 몇겹으로 되어 있으면 가장 안쪽의 괄호가 최우선이다. 예를 들면

```
12/(3+(1+2)^2)
```

이 경우의 가장 안쪽의 괄호안이 처리(1+2=3)되어서 다음과 같이 된다.

```
12/(3+3^2)
```

여기서 3 + 3 ∧ 2 는 지수연산이 먼저 실행되어 3 + 9 로 12가 되어 이것은 12 /12가 되므로 답은 1 이다.

지금까지 직접명령을 실행시켜 왔다. 그러나 스테이트먼트를 스토어해 두고 나중에 실행하는 간접실행에 대해서 알아 보자.

RUN

프로그램이 아무것도 들어 있지 않은 것을 분명히 하기 위해

```
]NEW
```

라고 타이프하고 RETURN 키를 치자. 지금까지 보아온 것처럼 NEW도 반드시 RETURN 에 의해 실행된다.

컴퓨터에 스테이트먼트를 넣어두게 하기 위해서는 스테이트먼트앞에 번호를 매겨 두자. 예를 들면

```
]10 PRINT 3+4
```

라고 타이프하고 RETURN 키를 눌러도 아무일도 일어나지 않는다. 즉 스테이트먼트를 스토어한 것이다. 그것을 보려면

```
]LIST
```

라고 타이프하고 RETURN 키를 누르면 미스타이프만 없으면

```
10  PRINT 3 + 4
```

가 스크린에 나타날 것이다. 그러면

```
]RUN
```

이라고 타이프 해 보자.

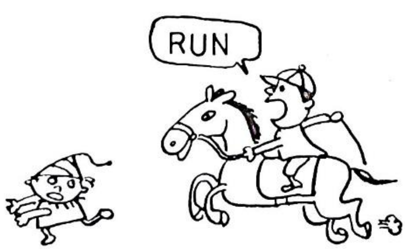
```
7
```

이라고 답이 화면에 나타난다. 이와같이 RUN이라고 타이프하면 스토어된 스테이트먼트가 실행된다.

그러나 아직 컴퓨터는 스테이트먼트를 잊은 것은 아니다. 몇 번이고 스테이트먼트를 RUN시킬 수 있다.

HOME

또 스크린을 클리어(clear)해도 스토어된 스테이트먼트는 잊어지지 않는다. 스



RUN



HOME

크린을 클리어 해 보자.

```
]HOME
```

HOME 커맨트는

ESC　SHIFT　@ p

와 똑같은 효과가 있는데, HOME은 직접실행이나 간접실행의 어느 쪽에서도 사용된다.

```
]100 HOME
```

이라고 타이프하고 RUN시켜보면 스크린을 클리어 시킨다.

```
]NEW
```

NEW

```
]LIST


]
```

NEW후 LIST를 해보면 프롬프트만 나타날 것이다. 즉 NEW라고 타이프하면 스토어되어 있던 스테이트먼트는 사라져 다시는 나타나지 않는다.

```
]RUN
```

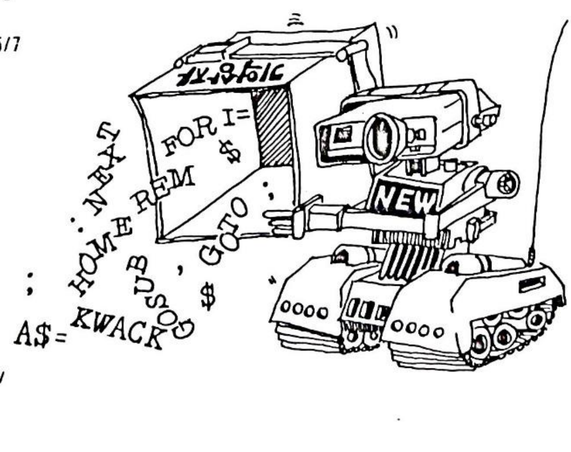
이라고 타이프해도 화면에는 아무것도 나타나지 않는다. 이것은 이전의 스테이트먼트가 NEW커맨트에 의해 소거되었기 때문이다.

그럼 각각 다른 번호를 붙여서 많은 스테이트먼트를 스토어 해 보자. LIST

```
]1 PRINT "APPLE"

]2 PRINT 3^5

]3 PRINT 35/7
```



NEW

를 타이프하고 LIST후 RUN시키면 그림과 같이 나타날 것이다.

```
* PROGRAM  1.2

]LIST

1  PRINT "APPLE"
2  PRINT 3 ^ 5
3  PRINT 35 / 7

]RUN
APPLE
243
5
```

그림1－3

컴퓨터에 스토어하기 위해 스테이트먼트 앞에 붙인 번호를 **행번호**라고 부른다 컴퓨터는 행번호가 작은 쪽부터 스테이트먼트를 스토어하고 실행한다. 실제로 이것을 확인하려면 앞의 스테이트먼트를 지우기 위해

```
]NEW
```

라고 타이프하고 다음의 스테이트먼트를 입력한다.

```
]0 PRINT "A"

]30 PRINT "E"

]20 PRINT "L"

]10 PRINT "P"
```

0 도 행번호로서 허용되어 있다. 또 행번호는 63999 까지 붙여진다.



LIST

RUN해 보면

```
]RUN
A
P
L
E
```

행번호를 아무렇게 붙여놓아도 단번에 순서대로 처리됨을 알았다. 컴퓨터 내부에서 무엇이 일어났는가를 보기위해

```
]LIST
```

라고 타이프하면

```
* PROGRAM 1.3
0  PRINT "A"
10  PRINT "P"
20  PRINT "L"
30  PRINT "E"
```

처럼 틀림없이 행번호순으로 스토어하고 있다.

이상에서 명령을 실행하기 전에 반드시 LIST를 취할 필요는 없지만 그렇게 하는 것도 좋은 방법이다.

RUN이라고 타이프했을 때에 실행되는 명령의 모임을 **프로그램**(program)이라고 한다.

프로그램을 횡으로 APPLE이 인자되게 실행시키고 싶다면 행번호사이에 기능을 추가하는 명령을 추가삽입시켜 프로그램을 개량할 수 있다.

즉 행번호 10과 20사이에

```
]15 PRINT "P"
```

를 추가하면 된다. 따라서 이렇게 행번호간의 간격을 충분히 벌려 두므로서 다시 타이프하는 번거로움을 없앨수 있을 것이다.



행번호

실행하기 전에 스토어 되었는지 LIST를 해 보자.

여기서 LIST는 요구된 모든 프로그램을 나타내며, 틀렸는지 맞았는지 요구하는 행번호의 스테이트먼트를 보기위해 다음의 여러가지 LIST문을 사용할 수 있다.

```
]LIST-20

0 PRINT "A"
10  PRINT "P"
15  PRINT "P"
20  PRINT "L"
```

은 행번호 20까지의 프로그램을 표시하며

```
]LIST20-

20  PRINT "L"
30  PRINT "E"
```

은 행번호 20부터 마지막 스토어된 행번호까지의 프로그램을 표시하고

```
]LIST15-20

15  PRINT "P"
20  PRINT "L"
```

은 행번호 15에서 행번호 20까지의 프로그램을 표시하고

```
]LIST10

10  PRINT "P"
```

은 행번호 10만 LIST한다.

LET

프로그램에서 데이타나 정보를 입력시키는 방법에는 여러가지가 있는데 그 중 하나가 LET문이다. LET는 "…이라 하자"는 의미의 대입문으로 그 일반형은

행번호 **LET** 변수명=산술식

이다. 예로서 A=3인 변수A의 값을 LET문을 사용하여 프린트하는 프로그램을 보자.

```
* PROGRAM  1.4
  10  LET A = 3
  20  PRINT A

  ]RUN
  3
```

여기서 LET를 생략하여도 된다.

```
* PROGRAM  1.5
10 A = 3
20  PRINT A

]RUN
3
```

결과는 마찬가지이다. 앞으로 이 책에서는 대부분 LET를 생략한다.
한개의 명령문속에 많은 변수의 값을 대입할 수 있는

```
10 A4 = B1 = C7 = D6 = 0
```

이런 것을 **다중LET문**(multiple LET statement)이라 한다.
또 한개의 명령문속에 2개이상의 명령을 합쳐놓는 것을 다중명령이라고 하며 꼭 LET문에서만 적용되는 것은 아니다. 앞으로 편리하겠기에 알아보자.

```
10 A = 3
20 B = 2
```

는 다중명령으로

```
10 A = 3:B = 2
```

로 각각 다른 행번호를 가진 스테이트먼트를 콜론(:)으로 연결하여 한데 묶을수 있다.
다음의 프로그램을 보자

```
* PROGRAM  1.6
10 A = 3:B = 2
20 C = 5
30  PRINT A + B + C
40  PRINT A * B
```

이 프로그램에서 변수A와 관계있는 행을 소거하려면 각각 소거하고자 하는 행번호를 타이프하고 [RETURN]를 누르면 삭제된다. 즉 20번 문만을 지우고자 할때

```
]20
```

이라고 타이프하고 [RETURN]키를 누른다.
물론 DEL(delete)명령을 사용할 수 있다. 그러나 Apple에서는 단지 한행만을 소거할때는 DEL명령을 이용할 수 없다. 만일 사용했을 경우 **DEL**

```
]DEL30

?SYNTAX ERROR
]
```

와 같은 에러메세지가 표시된다. 단, DEL은 프로그램의 블록을 소거하는데 사용된다.

```
]DEL20,30
```

이것은 행번호 20부터30까지의 모든 스테이트먼트를 소거한다. 큰 프로그램 등을 사용할 때 행번호를 블록마다 소거할 수 있는 DEL기능은 편리하다.

TRACE

프로그램의 흐름을 보는데 독특한 방법이 있다.

TRACE라는 것이 그것이다. 이것으로 프로그램이 어떻게 실행되는지 또는 실행중에 에러메세지가 나올 경우의 원인을 파악하는데 큰 도움을 줄 것이다.

```
]TRACE          RETURN
```

이 명령은 프로그램이 실행되는 순서에 따라 행번호를 나타내어 추적(追跡)할 수 있게 한다.

```
* PROGRAM  1.7
10 A = 5
20  LET Z = 7
30  PRINT Z,Z - A

]TRACE                  ←프로그램의 추적

]RUN
#10 #20 #30 7   2
```

위의 프로그램을 RUN시킬때 나타나는 #10, #20등은 행번호10, 20을 나타낸다. 즉 이 프로그램의 실행방식은 행번호10, 20을 행하고 행번호 30에서 Z는 7, Z−A＝7−5＝2을 수행하여 7과 2를 프린트하게 되었음을 알 수 있다.

여기서 TRACE는 다이렉트커맨드로서 사용되었다. 따라서 RUN시키면 프로그램은 모두 TRACE된다. TRACE는 행번호를 붙이지 않으면 프로그램 전체를 TRACE하고 또한번 프로그램을 RUN할때마다 TRACE가 걸린다. 이것을 멈추려면 프로그램중 또는 다이렉트 커맨트로

NOTRACE

```
]NOTRACE
```

를 실행한다.



DEL

NOTRACE명령은 TRACE명령을 취소(取消)시킨다.

SPEED=

이러한 모든 것들이 빠르게 실행될때 그것이 어떻게 되는가를 쉽게 알지 못하게 되는데 이경우 입출력 장치에 문자를 보내는 속도를 조절하기 위해서 SPEED 명령을 사용할 수 있다.

이때 가장 느린 것이 0이고, 가장 빠른 것이 255인데 일반적으로 이 명령을 사용하지 않을 때는 항상 255이다. 속도를 늦게하면 모니터상에 표시되는 문자도 늦게 표시된다.

```
* PROGRAM  1.8
  10  SPEED= 30: PRINT "ABCDEFGHI"
  20  SPEED= 255: PRINT "123456789"

  30  GOTO 10
```





[;]과[,]의 역할

참 고 PRINT문과 부호

①PRINT문과 PRINT문 사이에 콜론(:)이 있는 경우 : 행번호 20의 문이 두개와 같으므로 PRINT A와 PRINT B는 서로 다른 행에 인쇄

```
10 A = 10:B = 20
20  PRINT A: PRINT B
25 C = A + B
30  PRINT C
99  END

]RUN
10
20
30
```

②PRINT문 끝에 부호가 없는 경우 : 지시된 인쇄가 끝나면 행을 바꾼다.

```
10 A = 10:B = 20
20  PRINT A,B
25 C = A + B
30  PRINT C
99  END

]RUN
10                 20
30
```

③ PRINT문 끝에 커머(,)가 있는 경우 : 지시된 인쇄가 끝나도 같은 행 지정된 위치에 계속 인쇄

```
10 A = 10:B = 20
20  PRINT A,B,
25 C = A + B
30  PRINT C
99  END

]RUN
10                20                30
```

④ PRINT문 끝에 세미콜론(;)이 있는 경우 : 계속되는 출력이 그다음까지에 연이어 인쇄. 이것은 한개의 PRINT문으로 쓰는 것과 같은 효과.

```
10 A = 10:B = 20
20  PRINT A;B;
25 C = A + B
30  PRINT C
99  END

]RUN
102030
```

⑤ ④의 형태에 스페이스를 두기 위한 경우 : 계속되는 출력에서 구별지어 인쇄하기 위해 그 사이에 스페이스를 둠.

```
10 A = 10:B = 20
20  PRINT A;" ";B;" ";
25 C = A + B
30  PRINT C
99  END

]RUN
10 20 30
```

문제 1. 다음 프로그램은 PRINT문에 의해 어떠한 출력이 얻어지겠는가?

```
10 A = 100
20 B = 200
30 C = 300
40  PRINT A * B,B ^ 2,C - A / C
50  PRINT C + (A + B * 3) - A
60  PRINT A ^ 0.5,A ^ 1 / 2
```

```
]RUN
20000           40000           299.666667
900
10              50
```

문제 2. 126−72를 구하는 프로그램을 작성하라.

```
10  LET M = 126 - 72
20  PRINT M

]RUN
54
```

문제 3. (57+42) * (17−8)를 구하는 프로그램을 작성하라.

```
10  LET A = (57 + 42) * (17 - 8)
20  PRINT A

]RUN
891
```

문제 4. 64+19*7−17을 구하는 프로그램을 작성하라.

```
10  LET A = 64 + 19 * 7 - 17
20  PRINT A

]RUN
180
```

문제 5. 아래의 것을 인쇄하는 프로그램을 작성하라.

```
]RUN
*
 *
  *
****
```

```
10  PRINT "*"
20  PRINT " *"
30  PRINT "  *"
40  PRINT "****"
```

문제 6. 다음의 계산을 수행하는 프로그램을 작성하고 그 결과를 손으로 계산하여 확인해 보라.

(a) $8 \times (19-4)$

(b) $1\times2+2\times3+3\times4+4\times5$

(c) $1+1\div10+1\div100$

(d) $10-20+30-40+50-60$

```
10  PRINT 8 * (19 - 4)
20  PRINT 1 * 2 + 2 * 3 + 3 * 4 +
     4 * 5
30  PRINT 1 + 1 / 10 + 1 / 100
40  PRINT 10 - 20 + 30 - 40 + 50 -
     60

]RUN
120
40
1.11
-30
```

문제 7. 아래 프로그램의 정확한 출력은 무엇인가?

```
10 A = 3
20  PRINT "THE VALUE OF B"
30 B = A + 4 * 4
40  PRINT B
```

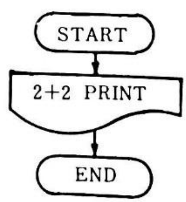
```
]RUN
THE VALUE OF B
19
```

문제 8. X= 5 일때 Y= 5X+ 7이면 Y의 값이 얼마인가를 나타내는 프로그램을 작성하라.

```
10 X = 5
20 Y = 5 * X + 7
30  PRINT Y

]RUN
32
```

문제 9. 2+2라는 식을 계산해서 그 결과를 인쇄하기 위한 흐름도와 프로그램을 작성하라.




```
10  PRINT 2 + 2
20  END

]RUN
4
```

> **문제** 10. A팀의 5시합의 득점은 3, 5, 0, 1, 2 점이었다. A 팀의 1 시합 평균 득점능력을 구하는 프로그램을 작성하라.

```
10 A = (3 + 5 + 0 + 2 + 1) / 5
20  PRINT "A TEAM'S AVERAGE SCORE
      ";A
30  END

]RUN
A TEAM'S AVERAGE SCORE 2.2
```

1 – 3 데이타(DATA)의 입력

INPUT

```
]10 INPUT A
```

프로그램이 이 명령을 실행하면 스크린상에 ?가 표시된다. 여기서 키보드로부터의 입력을 기다라고 있으므로 소정의 값을 키인해서 RETURN 으로 하면 A의 값이 정해지고, 프로그램은 실행을 계속한다.

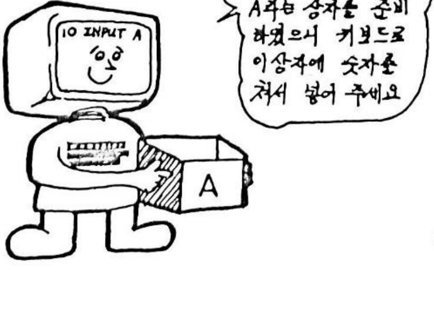
여기서 컴퓨터가 친구에게 말을 거는 것같이 하면, 어렵지 않게 키인할 수 있으므로 다음과 같이 PRINT문을 넣을 수 있다.

```
]10 INPUT "A= ";A
```

INPUT문 중에서의 메세지는 PRINT문과 마찬가지로 쿼트로 둘러 쌀 필요가 있다. 그리고 메세지와 변수명 사이에는 세미콜론(;)이 필요하다. INPUT 문에 메세지가 섞여 있을 때 그 뒤에 ?는 나오지 않는다. 만약 ?를 붙이려면 메세지 중 ?를 포함해 두어야 한다.

```
]10 INPUT "A=? ";A
```

만약 입력한 수치를 틀리게 되었을 때는 RETURN 를 누르기전이면 우측 또는 좌측 화살표 →, ← 를 사용해서 커서를 움직여 다시 키인할 수 있다. 그러나 수행하기 전에 RETURN 를 누르면 에러메세지가 나타난다. 또 숫자가 아니고, 잘못해서 알파벳을 눌렀을 경우에도



INPUT

```
]RUN
?
?REENTER

]RUN
A= J
?REENTER
```

라고 나오면서 ?가 나타나므로 다시 한번 더 키인한다.

또 INPUT문으로 몇개의 값을 동시에 넣을 수도 있다.

```
]100 INPUT X,Y,Z
```

? 마크를 내고 입력을 기다리는 것은 같으므로

```
?3,4,5
```

와 같이 소정의 값을 3개의 커머(,)로 구획지어 키인하고 RETURN 을 누르면 처음의 것이 X에, 다음에 Y, 최후의 변수 Z에 스토어된다.

다음과 같이 수를 넣을 때마다 RETURN 이라도 상관없다.

```
]RUN
?3
??4
??5
```

3개를 모두 넣으면 다음으로 진행된다.

GET

INPUT명령에서는 스크린에 나타내면서 데이타를 받아들이는데 GET문은 화면에 나타내지 않고 또 RETURN 키를 누르지 않은 상태로 키보드로 부터 한글자씩 입력해 들인다.

```
* PROGRAM 1.9
]10 GET X$

]20 IF X$="0" THEN END

]30 PRINT X$

]40 GOTO 10

]RUN
A
B
C
D
```

A, B, C, D를 입력시키고 다섯번째 0을 넣은 경우의 결과이다.

READ, DATA

또 프로그램중에 그렇게 자주 변경은 없지만 변경할 필요가 있을 때에는 간단히 변경될 수 있는 상수를 삽입해 두고 싶다고 하자. 이럴때 사용하는 것이 RE

AD, DATA문이다.

```
* PROGRAM  1.10
]10 READ A,B,C,D

]20 X=A+B+C+D

]30 PRINT X

]40 DATA 3,5,7,9
```

이 프로그램은 READ문에 의해 DATA문 중의 수치를 꺼내어 계산을 해 가는 것으로 READ문의 동작은 기본적으로는 INPUT문과 같으며 입력선이 키보드에서 프로그램내의 DATA문으로 옮길 뿐이다.

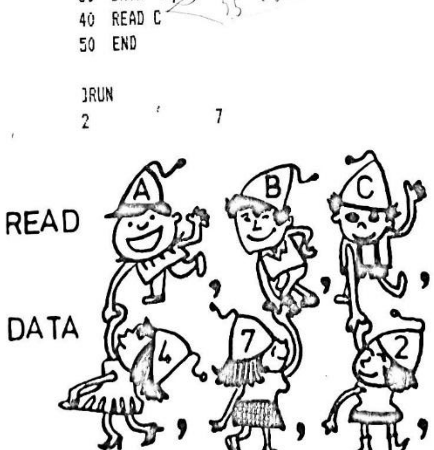
프로그램이 실행되어서 DATA문과 마주치면 컴퓨터는 프로그램중에 있는 DATA문의 최초의 데이타를 꺼내어 지정된 변수에 대입한다.

다음의 READ문에 대해서는 DATA문의 2 번째 데이타, 3 번째의 READ문에 대해서는 3 번째의 데이타라는 식으로 계속해가며 DATA문이 끝나면 프로그램은 끝이 난다. DATA문은 1 행에 모아서 쓸 필요는 없고 프로그램 중의 1 개소에 모아둘 필요도 없다. 따라서 여러개의 행으로 나누어도 된다.

그러나 다음 프로그램과 같이 READ문의 3 가지 A, B, C에 비해 DATA 문은 데이타가 2 개로 READ문의 C에 대응되는 데이타가 없을 때에는 에러메세지를 낸다.

```
* PROGRAM  1.11
10  READ A,B
20  PRINT A,B
30  DATA  2,7
40  READ C
50  END

]RUN
2              7
```



READ, DATA

```
?OUT OF DATA ERROR IN 40
]
```

이것을 막는 것이 RESTORE이다.

RESTORE는 DATA문의 선두로 되돌아 가도록 한다. 앞서의 프로그램에

```
]35 RESTORE          ]45 PRINT C
```

를 첨가해 보자.

입력한 수가 DATA문 중에 없으면 에러메세지를 내지만, 이 경우 행번호 35에서 RESTORE문을 사용해서 DATA 리스트포인트를 제일 먼저 되돌려 READ문의 C값에 2가 대입되어 수행하게 된다. 즉 RESTORE 명령은 프로그램의 최초 DATA 명령의 첫 데이타부터 순차로 읽어들이도록 해 준다.

RESTORE

```
10  READ A,B
20  PRINT A,B
30  DATA  2,7
35  RESTORE
40  READ C
45  PRINT C
50  END

]RUN
2              7
2
```

이외에 데이타를 입력하는데는 이미 설명한 LET문을 사용할 수 있으며 이것은 수치의 대입을 행할 수 있다.

REM

이러한 프로그램을 작성할 경우 프로그램 중에 약속이나 코멘트, 설명을 삽입시켜 어떤 과정의 프로그램인지를 알 필요가 있는데 이때는 REM문을 사용한다. REM이란 것은 Remark의 약어로 프로그램 중의 코멘트나 설명을 삽입하기 위해 사용된다. 실행시에는 완전히 무시되므로 무엇을 써도 상관없다.

즉 REM은 어떠한 종류의 문자, 주로 주석을 위한 문자도 프로그램 중에 삽입하는 것을 가능하게 한다. REM은 RETURN에 의해서만 그 종료를 표시할 수 있다.

REM문은 REM뒤에 인자된 문자의 수에 관계없이 LIST때에는 어느 일정한 스페이스가 REM직후에 넣어진다.

```
* PROGRAM  1.12
 ]0 REM ********************

 ]1 REM *    R  E  M      *

 ]2 REM ********************

 ]LIST0-2

 0  REM  ********************
 1  REM  *    R  E  M      *
 2  REM  ********************
```

문제 11. INPUT문을 이용하여 주어진 길이, 폭, 높이에 의해 체적을 구하는 프로그램을 작성하라.

```
10  INPUT "LENGTH,WIDTH,& HEIGHT?
      ";L,W,H
20  PRINT "VOLUME IS ";L * W * H

]RUN
LENGTH,WIDTH,& HEIGHT? 4,5,6
VOLUME IS 120
```

12. 가격P는 센트로 표시되는 빵의 단가이고 갯수N은 빵의 수를 입력시킬 때, 전체의 가격이 달러$로 표시되도록 프로그램을 작성하라.

```
10  INPUT "PRICE,NUMBER OF LOAVES
     ? ";P,N
20  PRINT "TOTAL SPENT = $";P * N
     / 100

]RUN
PRICE,NUMBER OF LOAVES? 52,7
TOTAL SPENT = $3.64
```

13. 삼각형의 밑변과 높이를 입력시켜 넓이를 구하는 프로그램을 작성하라. 단, 삼각형의 넓이를 프린트하기 전에 한 줄을 띄우라.

```
10  INPUT "WHAT IS THE BASE? ";B
20  INPUT "WHAT IS THE ALTITUDE?
     ";A
30 T = A * B * 0.5
40  PRINT
50  PRINT "THE AREA IS ";T

]RUN
WHAT IS THE BASE? 6
WHAT IS THE ALTITUDE? 8

THE AREA IS 24
```

14. 대출도서의 과태료를 계산하자. 대출 도서의 권수와 지연일수를 입력했을 때, 권당 일일 과태료가 10¢(센트)일 경우 총 과태료를 인쇄하는 프로그램을 작성하라.

```
10  INPUT "HOW MANY BOOKS HAVE YO
     U BOLLOWED? ";B
20  INPUT "HOW MANY DAYS LATE ARE
     THEY? ";L
30  PRINT "YOU OWE ";.10 * B * L;
     " DOLLARS."

]RUN
HOW MANY BOOKS HAVE YOU BOLLOWED? 5
HOW MANY DAYS LATE ARE THEY? 30
YOU OWE 15 DOLLARS.
```

15. 프로선수들은 신중한 협상을 통해 놀랄만한 수입을 올리고 있다. 물론 진정한 승리자는 홍정치 않는 Uncle Sam이지만. 자, 선수이름과 수입을 물어보고 만약 그의 수입에 해당하는 소득세율이 44% 이상이면 그 선수의 성명, 실질소득, 세금을 인쇄하는 프로그램을 작성하라.

```
10  INPUT "WHAT IS THE PLAYER'S N
     AME? ";P$
20  PRINT "WHAT IS ";P$;"'S ";
30  INPUT "WAGE? ";W
40 T = W * .44
50  PRINT P$;" WOULD KEEP $";W -
     T
60  PRINT "HE WOULD PAY $";T;" IN
      TAXES."

]RUN
WHAT IS THE PLAYER'S NAME? KWACK
WHAT IS KWACK'S WAGE? 24
KWACK WOULD KEEP $13.44
HE WOULD PAY $10.56 IN TAXES.
```

16. 51, 5, 9, 8의 데이타를 읽고 100으로 부터 차례로 51, 5, 9, 8순으로 모두 뺄셈을 하고 그 답을 인쇄하는 프로그램을 작성하라.

```
10 A = 100
20 READ X1
```

```
30 A = A - X1
40  READ X2,X3
50 A = A - X2 - X3
60  READ X4
70 A = A - X4
80  PRINT "100-";X1;"-";X2;"-";X3
    ;"-";X4;"= ";A
90  DATA  51
100  DATA  5,9,8
110  END

]RUN
100-51-5-9-8= 27
```

17. 아래의 데이타를 읽고 그 제곱값을 인쇄하는 프로그램을 작성하라.

6, 1.5, 12

```
10  READ A,B,C
20  PRINT A;"^2=";A ^ 2
30  PRINT B;"^2=";B ^ 2
40  PRINT C;"^2=";C ^ 2
50  DATA  6,1.5,12
60  END

]RUN
6^2=36
1.5^2=2.25
12^2=144
```

18. 9,6,8,4,3등 5개의 데이타를 읽고 합계를 구하는 프로그램을 작성하라.

```
10  READ A,B,C,D,E
20 ANS = A + B + C + D + E
30  PRINT A;"+";B;"+";C;"+";D;"+"
    ;E;"= ";ANS
40  DATA  9,6,8,4,3

]RUN
9+6+8+4+3= 30
```

19. 데이타 A, B를 입력해서 A와 B의 가감승제를 구하라.

P＝A＋B,　　Q＝A－B,

R＝A＊B,　　S＝A／B

```
10  INPUT A,B
20 P = A + B
30 Q = A - B
40 R = A * B
50 S = A / B
60  PRINT "P= ";P
70  PRINT "Q= ";Q
80  PRINT "R= ";R
90  PRINT "S= ";S
100  GOTO 10
110  END

]RUN
?15,5
P= 20
Q= 10
R= 75
S= 3
?
```

20. 수를 2개 입력하여 4X＋5Y를 구하는 프로그램을 작성하라.

```
10  INPUT "X=";X
20  INPUT "Y=";Y
30 Z = 4 * X + 5 * Y
40  PRINT Z
50  END

]RUN
X=32
Y=500
2628
```

21. 5명이 체력측정을 하였다. 각각의 턱걸이 회수를 입력하고 평균값을 구하는 프로그램을 작성하라.

```
10  INPUT A,B,C,D,E
20 F = (A + B + C + D + E) / 5
30  PRINT A;" ";B;" ";C;" ";D;" "
     ;E
40  PRINT "AVERAGE=";F
50  END

]RUN
?12,16,8,17,23
12 16 8 17 23
AVERAGE=15.2
```

문제22. Y＝3X＋5 이고 X가 3, 5, 12, 17, 8 일때 Y의 값을 구하기 위해 READ, DATA 문을 사용하여 프로그램을 작성하라.

```
10  READ X
20 Y = 3 * X + 5
30  PRINT Y
40  GOTO 10
50  DATA  3,5,12,17,8

]RUN
14
20
41
56
29

?OUT OF DATA ERROR IN 10
```

문제23. 이름과 각 학생의 맨처음 점수만을 나타내는 DATA 문의 프로그램을 작성하라.

```
10  PRINT "NAME","1ST GRADE"
20  READ N$,A,B,C,D
30  PRINT N$,A
40  GOTO 20
50  DATA   KIM,83,75,86,70,HONG,7
      9,86,89,77,YOON,92,78,83,73

]RUN
NAME            1ST GRADE
KIM             83
HONG            79
YOON            92

?OUT OF DATA ERROR IN 20
```

문제24. 세 학생의 성적과 이름을 읽고 아래와 같이 이름과 평균을 출력하는 프로그램을 작성하라.

```
]RUN
NAME            AVERAGE
KIM             78.5
HONG            82.75
YOON            81.5
```

```
10  PRINT "NAME","AVERAGE"
20  READ N$,A,B,C,D
25 X = (A + B + C + D) / 4
30  PRINT N$,X
40  GOTO 20
50  DATA   KIM,83,75,86,70,HONG,7
      9,86,89,77,YOON,92,78,83,73
```

문제25. 다음 프로그램의 출력을 머리로 계산하여 써보고, 프로그램을 실행시킨 결과와 비교해 보라.

```
10  READ A,B,C,D
20  PRINT A,B
30  PRINT C;"    ";D
40  DATA  3E3,786,4E-1,.654
```

```
]RUN
3000            786
.4    .654
```

문제26. 당신이 하루에 8 시간 씩 잔다고 가정하고, 컴퓨터로 하여금 당신이 평생동안 취한 수면의 총시간수를 프린트하는 프

로그램을 작성하라. 당신의 출생일과 오늘 날짜를 년, 월, 일로 입력한다. 단, 한달은 30일, 일년은 365일로 한다.

```
10  INPUT "YEAR,MONTH,& DAY OF BI
     RTH? ";Y1,M1,D1
20 DB = Y1 * 365 + M1 * 30 + D1
30  INPUT "TODAY'S YEAR,MONTH,& D
     AY? ";Y2,M2,D2
40 DT = Y2 * 365 + M2 * 30 + D2
50 S = 8 * (DT - DB)
60  PRINT "YOU HAVE SLEPT ABOUT "
     ;S;" HOURS."

]RUN
YEAR,MONTH,& DAY OF BIRTH? 1963,11,4
TODAY'S YEAR,MONTH,& DAY?  1983,11,4
YOU HAVE SLEPT ABOUT 58400 HOURS.
```

문제 27. 다음과 같이 데이타가 주어졌을 때 2X+3Y+8Z라는 수식의 값을 인쇄하는 프로그램을 작성하라.

X	Y	Z
3	2	0.5
−7	2.5	10^2

```
10  READ X,Y,Z
20  PRINT 2 * X + 3 * Y + 8 * Z
30  GOTO 10
40  DATA  3,2,0.5,-7,2.5,1E2

]RUN
16
793.5

?OUT OF DATA ERROR IN 10
]
```

1-4 점프명령

비지니스 및 과학분야에서 응용되는 대부분의 프로그램은 다량의 자료를 처리하는 관계로 반복되는 프로그램이 필요하게 된다. 이때 가장 간단하게 사용할 수 있는 것이 GOTO문이다. GOTO

GOTO커맨드는 의미하는 바대로 ~로 점프해가는 **무조건 점프명령**이다.

```
* PROGRAM  1.13
 ]10 PRINT A

 ]20 GOTO 10
```

위의 프로그램은 GOTO문을 사용한 간단한 프로그램이다. 여기서 행번호 10은 A를 프린트하라고 표시한다. 행 20은 「행 10으로 가서 그곳을 실행하여라」라는 것이다. 즉 GOTO문은 흐름을 바꿔줄 행번호를 지정한다. 위 프로그램을 실행시키면 A를 프린트하고 행 20으로 가면 행 10으로 되돌아 가라고 한다. 영원히 이 반복이다. 물론 A의 변수에 대한 값이 주어있지 않으므로 0만 반복될 것이다.

이런 반복이 **루프**이다. 루프는 한번 실행된 스테이트먼트로 다시 되돌아 오는 커맨드가 사용되고 있을 때 일어난다. 루프를 이용하면 필요한 프로그램의 수를

줄일 수 있기 때문에 대부분의 프로그램문 속에 포함되어 있다.

RUN하고 있는 이 프로그램을 멈추게 하려면

CTRL
C

로 하면 이 커맨드에서는 실행이 어디서 정지했는지를 인자한다.

```
BREAK IN 10
]
```

와 같이 어느 행 번호에서 멈췄는지를 알 수 있다. 이것은 모든 커맨드의 끝에 RETURN을 눌러야 한다는 룰의 유일한 예외이다. 보통 CTRL-C로 프로그램을 멈추게 했을 때는 RETURN 키는 필요없다.

또 RESET 키라도 프로그램을 멈추게 할 수 있지만 앞서와 같이 어느 행번호에서 멈추었다는 표시는 되지 않는다.

CTRL-C나 RESET 키로 프로그램을 멈추게 했을 때는 다시 계속하기 위해

```
]CONT
```

이것은 Continue를 말하며, CONT는 중지되었던 명령의 다음 명령부터 실행되도록 한다.

다음 프로그램을 보자.

```
* PROGRAM 1.14
 ]10 N=1

 ]20 PRINT N

 ]30 N=N+1

 ]40 GOTO 20
```



GOTO

이 프로그램도 마찬가지로 숫자가 1개씩 증가되면서 계속 실행을 한다. 이것을 어느 숫자에서 멈추도록 하려면 몇 번 루프를 돌릴 것인지를 정하여 컨트롤 하는 방법이 있다. 예를 들어 만일 N이 10미만이면 다시 GOTO되고 N이상이면 멈추게 하는 명령이다.

IF-GOTO

이러한 스테이트먼트를 IF문이라고 한다.

만약 조건이 충족되면 컴퓨터는 GOTO 명령을 뛰어넘어서 다음행의 명령을 실행한다. 만약 다음의 행이 없으면 프로그램은 끝이 난다.

1부터 9까지 인자하고 정지하는 프로그램은 다음과 같다.

```
* PROGRAM  1.15
  10 N = 1
  20  PRINT N
  30 N = N + 1
  40  IF N < 10 GOTO 20
```

일반적으로 IF문은 다음과 같이 작용한다.

IF 수식표현 GOTO 행번호

GOTO문에는 또 ON-GOTO문이 포함된다.

ON-GOTO

예를 들어 수A를 입력하고 1일때는 A=1, 2일때는 A=2로 인쇄하고 1과 2 이외에 대해서는 재입력하는 프로그램을 작성해보면 다음과 같으며 이러한 점프명령은 계산형GOTO문이다. 계산형 GOTO문은 복수의 분기를 가진 **조건부 점프명령**이다. 프로그램의 제어는 다른 점프명령과는 달리, 다수의 행번호 중의 한 개 행번호의 문으로 옮겨간다. 즉 다수의 IF문을 하나의 ON-GOTO로 처리된다.

```
* PROGRAM  1.16
  ]10 INPUT A                         ]RUN
                                      ?1
  ]20 ON A GOTO 40,60                 A=1

  ]30 GOTO 10                         ]RUN
                                      ?2
  ]40 PRINT " A=1 "                   A=2

  ]50 END                             ]RUN
                                      ?3
  ]60 PRINT " A=2 "                   ?4
                                      ?
  ]70 END                             ?REENTER
                                      ?
```

여기서 A가 1이면 행번호 20에서 GOTO40이 되어 행번호 40에서 A=1이라고 인자하고, 또 A가 2로 입력되면 GOTO60이 되어 행번호 60에서 A=2로 인자될 것이며, 만일 3이나 4로 2보다 큰 수 일때는 무조건 행번호 30에 의해 재입력하도록 ?가 다시 나타날 것이다.

이상과 같이 GOTO문은 분기가 자유로와 편리하지만 이것을 많이 써서 프로그램을 짜 놓으면 프로그램이 이해하기 곤란해지므로 빈번한 사용은 피해야 한다. 그러므로 GOTO문을 많이 쓴 프로그램은 잘못 작성된 프로그램이라 할 수 있다.

문제 28. X의 값을 입력하면 계산기가 5 * X와 X/5를 계산할수 있는 프로그램을 작성하라. 단 출력은 정확히 아래와 같이 나와야 한다.

```
]RUN
?56
5*X= 280
X/5= 11.2
?4
5*X= 20
X/5= .8
?
```

```
10  INPUT X
20  PRINT "5*X= ";5 * X
30  PRINT "X/5= ";X / 5
40  GOTO 10
```

문제 29. N! (factorial)을 구하는 프로그램을 작성하라.

```
10  REM  * FACTORIAL *
20 N = 1:X = 1
30 X = X * N
40  PRINT N;"! = ";X
50 N = N + 1
60  GOTO 30
70  END

]RUN
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
```

STOP

프로그램을 작성하여 RUN시켜도 한 번만에 생각대로 실행되는 것은 아니다. 그런 때에는 적당한 곳에 STOP명령을 프로그램 해놓고, 정지했을 때의 변수가 어떤 내용으로 되어 있는가 확인하고 싶을 때가 있다. 예를 들면 다음과 같은 프로그램에서는

```
* PROGRAM  1.17
 10  READ A,B
 20 X = A * B
 30  STOP
 40 Y = A / B
 50  PRINT X,Y
 60  DATA  50,5
 70  END

 ]RUN

 BREAK IN 30
```

로 된다. 이때 직접 실행 방식으로써

```
]PRINT A,B,X
50            5             250
```

로 변수를 표시시켜서 프로그램의 검사가 된다. 여기서 프로그램을 다시 시작하고 싶을 때는

CONT

```
]CONT
250           10

]
```

로 프로그램을 끝까지 수행시킨다.

다음의 프로그램은 행이 변할수록 「*」기호가 하나씩 늘어 피라밋 형태를 표시하는 것이다.

```
* PROGRAM  1.18
  10 A = 0:B = 38:C = 1
  20  FOR X = A TO B STEP C
  30  FOR Y = 0 TO X
  40  PRINT "*";
  50  NEXT Y: PRINT : NEXT X
  60 K = A:A = B:B = K
  70 C =  - C: GOTO 20
```

이 프로그램을 RUN시켜 수행시키면

```
]RUN
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
*************
**************
***************
****************
*****************
```



STOP, CONT

여기서 [RESET] 키(혹은 [SHIFT] - [RESET])를 누르면 실행을 중지한다. 여기에 이번에는

```
]100 END
```

로 프로그램을 추가하고서

```
]CONT
```

명령을 입력시키면

```
?CAN'T CONTINUE ERROR
]
```

그렇다. STOP명령 또는 [RESET] 키로서 명령을 중지한 후에 행번호를 사용한 프로그램의 편집을 하면, CONT명령은 사용할 수 없음을 알았다.

> **문제 30.** A= 1, B= 2, C= 3, D = 4, E= 5의 수를 두고 그 값들이 바로 입력되었는지를 각수를 인쇄하고, 프로그램을 일단 정지시켜 조사하여 모든수의 합을 계산하는 프로그램을 작성하라.

```
10 A = 1:B = 2:C = 3:D = 4:E = 5
20 T = A + B + C + D + E
30  PRINT A,B,C,D,E
40  STOP
50  PRINT "TOTAL=";T
60  END

]RUN
1               2               3
4               5

BREAK IN 40
]CONT
TOTAL=15
```

> **문제 31.** 가로 9, 세로 5인 사각형의 면적을 인쇄하고, 프로그램을 정지시켜 그 값이 맞는가 확인하고 다시 밑변이 9, 높이가 5인 삼각형의 면적을 구하도록 프로그램을 작성하라.

```
10 A = 9
20 H = 5
30 S = A * H
40  PRINT S
50  STOP
60 SS = A * H / 2
70  PRINT SS
80  END

]RUN
45

BREAK IN 50
]CONT
22.5
```

1-5 의사결정과 루프

IF-THEN

컴퓨터는 인간의 경우와 같이 판단이라는 중요한 기능을 가지고 있다. 다시말

해 모든 컴퓨터 언어에 갖추어져 있는 강력하고 기본적인 기능은 계산의 흐름을 결정하기 위해 값을 비교하기도 하고 검사하기도 하는 것이다. 즉 어떤 조건이 만족될 때에만 일종의 GOTO문이 일시적으로 준비된다.

이것은 컴퓨터에 있어 가장 중요한 기능중의 하나이다.

IF-THEN문은 단순한 계산이 아니라 컴퓨터의 가장 중요한 기능인 "판단한다"고 하는 동작을 수행하는 명령문이다. 따라서 IF-THEN문은 조건점프를 가능하게 하는데, 즉 프로그램이 실행중에 생기는 조건에 따라 자기 자신의 처리의 흐름을 결정하도록 하고 있다.

IF-THEN문을 사용하여 입력된 숫자가 0인지 아닌지를 알아보는 프로그램을 만들어 보자.

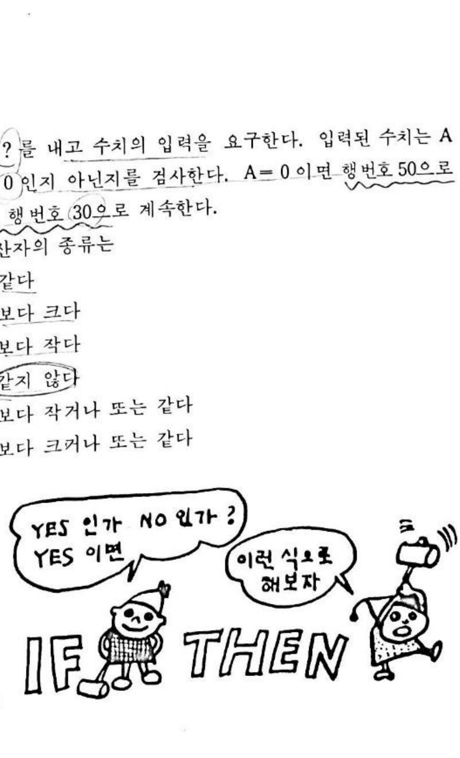
```
* PROGRAM 1.19
]10 INPUT A
]20 IF A=0 THEN GOTO 50
]30 PRINT " NO"
]40 GOTO 10
]50 PRINT " YES"
]60 GOTO 10
```

이 프로그램은 RUN하면 ?를 내고 수치의 입력을 요구한다. 입력된 수치는 A에 대입되고 행번호 20에서 0인지 아닌지를 검사한다. $A=0$이면 행번호 50으로 점프하고 $A \neq 0$이면 다음의 행번호 30으로 계속한다.

조건 판단에 사용되는 연산자의 종류는

=	같다
>	보다 크다
<	보다 작다
<>또는 ><	같지 않다
<= 또는 =<	보다 작거나 또는 같다
>= 또는 =>	보다 크거나 또는 같다

의 6종류가 있다.



IF-THEN

여기서 A, B 2개의 값중 큰쪽을 판단해서 표시하는 프로그램을 만들어 보자. 물론 지금까지 실행한 프로그램을 지우기 위해 NEW를 실행하는 것을 잊지 말아라.

* PROGRAM 1.20

```
10  INPUT A,B
20  IF A <  = B THEN  GOTO 50
30  PRINT "A>B"
40  GOTO 10
50  IF A < B THEN  GOTO 80
60  PRINT "A=B"
70  GOTO 10
80  PRINT "A<B"
90  GOTO 10
```

```
]RUN
?234,567
A<B
?345,345
A=B
?897,231
A>B
?-456,-345
A<B
?567,-567
```

이 프로그램을 실행하면, 행번호 10에서 커머로 구획지어진 수치를 2개 요구하고 입력A, B에 대입된다. 행번호 20에서 「A가 B보다 작거나 또는 같다」라는 사항을 체크하고 이 조건을 만족시키고 있을 때는 다시 「A=B」의 조건을 체크하기 위해서 행번호 50으로 점프한다.

행번호 30은 행번호 20의 조건을 만족시키지 않을 때만 프로그램이 지나가므로 여기서 A>B라는 표시를 한다.

행번호 50에서는 「A<B」를 체크하고 조건에 만족되지 않을 때는 행번호 60에서, 「A=B」로 만족되어 있을 때에는 행번호 70으로 분기한다.

이러한 IF-THEN문은 아래와 같이 여러가지 형태로 사용할 수 있다.

즉…의 조건이 眞(참)이면 THEN이하의 명령이 실행되고 僞(거짓)이면 다음 행번호로 진행한다.

IF…THEN GOTO 또는 IF…GOTO
IF…PRINT 또는 IF…THEN PRINT
IF…THEN A=3 * 2 같은 대입문
IF…THEN INPUT
IF…THEN READ
IF…THEN GOSUB
IF…THEN RETURN
IF…THEN STOP
IF…THEN END

AND, OR

IF…THEN문장에서 조건이 여러개 생각될 때 문장 수식어 AND와OR가 사용된다.

다음 문의 예에서

```
20  IF X > 5 AND Y = 3 THEN 50
```

는 두가지 조건이 모두 사실일 때만 행번호50으로 이전된다. 다음 문

```
20  IF X > 5 OR Y = 3 THEN 50
```

는 두가지 조건중에서 하나만이라도 사실이면 프로그램이 행번호50으로 넘어 간다.

다음 프로그램은 OR수식어를 사용하여 지금까지 배운것을 정리해 보는 프로그램이다.

* PROGRAM 1.21

```
10  INPUT "WHO ARE YOU? ";N$
20  IF N$ = "JAMES BOND" OR N$ =
    "007" THEN 50
30  PRINT "THIS MISSION IS NOT ME
    ANT FOR YOU! "
40  PRINT : GOTO 10
50  PRINT "THESE ORDERS ARE FOR Y
    OUR EYES ONLY!"
60  INPUT "ARE YOU ALONE? ";A$
70  IF A$ < > "YES" THEN  PRINT
    "TRY LATER.": GOTO 10
80  PRINT : PRINT "   DR. GOODHEA
    D HAS ESCAPED! YOUR"
90  PRINT "MISSION, SHOULD YOU DE
    CIDE TO ACCEPT"
100 PRINT "IT, IS TO SEEK OUT GO
    ODHEAD AND"
110 PRINT "RETURN HER TO THE TEL
    ENGARD PRISON"
```

```
]RUN
WHO ARE YOU? HONG KIL DONG
THIS MISSION IS NOT MEANT FOR YOU!

WHO ARE YOU? 007
THESE ORDERS ARE FOR YOUR EYES ONLY!
ARE YOU ALONE? NO
TRY LATER.
WHO ARE YOU? JAMES BOND
THESE ORDERS ARE FOR YOUR EYES ONLY!
ARE YOU ALONE? YES

   DR. GOODHEAD HAS ESCAPED! YOUR
MISSION, SHOULD YOU DECIDE TO ACCEPT
IT, IS TO SEEK OUT GOODHEAD AND
RETURN HER TO THE TELENGARD PRISON
```

여기서 행번호10은 사용자의 이름을 묻는다. N$로 입력된 이름은 행번호20에서 두가지 이름과 비교된다. 만일 N$가 「JAMES BOND」거나 「007」이면 프로그램은 행번호50으로 뛰어 넘는다. 그렇지 않으면 행번호30에 의해 메세지가 인쇄되고 프로그램은 다시 행번호10으로 되돌아 간다. 행번호50에서 경고문이 인쇄되고 60에서 Bond가 혼자인지 아닌지를 물어온다. 여기서 행번호70에서 「YES」 이외의 대답을 하면 프로그램은

```
TRY LATER
```

를 인쇄하고 행번호10으로 되돌아 가게 된다. 그렇지 않으면 임무설명이 행번호 80~110에 걸쳐 인쇄된다.

END

END문은 프로그램의 실행을 정지시키는데 사용된다. 예로써

```
100 END
```

는 행번호100에 도달하면 프로그램은 그 실행이 중지된다. END문은 프로그램중에서 한 군데 이상 나와도 괜찮고 IF…THEN문에 넣어도 된다.

다음 프로그램은 END문을 사용하여 「KWACK」라는 이름이 입력되었을 때 프로그램 수행을 중지시킨다.

```
* PROGRAM  1.22
 10  INPUT "WHAT IS YOUR NAME? ";N
     $
 20  IF N$ = "KWACK" THEN  END
 30  PRINT "IT'S GOOD TO MEET YOU
     ";N$
 40  PRINT : GOTO 10

 ]RUN
 WHAT IS YOUR NAME? PARK
 IT'S GOOD TO MEET YOU PARK

 WHAT IS YOUR NAME? KWACK
```

문제 32. $1^2 - 2^2 + 3^2 - \cdots\cdots 100^2$ 의 합을 계산하라.

```
10 N = 0
20 S = - 1
30 I = 1
40 S = - S
50 N = N + S * I ^ 2:N% = N
60 I = I + 1
70  IF I < = 100 THEN 40
80  PRINT " ANSWER= ";N%
90  END

]RUN
 ANSWER= -5050
```

문제 33. $1 + \frac{1}{1!} + \frac{1}{2!} + \cdots\cdots + \frac{1}{10!}$ 의 합을 계산하라.

```
10 A = 1:F = 1
20 X = 1
30 F = F / X:A = A + F
40 X = X + 1
50  IF X <  = 10 THEN 30
60  PRINT " ANSWER= ";A
70  END

]RUN
 ANSWER= 2.7182818
```

문제 34. 50m 의 언덕길을 올라가려는 리어카가 있다. 그런데 이 리어카는 한 번에 3m를 미끄러져 내려온다. 언덕에 올라 가는데는 몇번이나 걸리겠는가 를 계산하는 프로그램을 작성하라.

```
10 R = 0:K = 0
20 R = R + 3:K = K + 1:R = R - 2
30  IF R < 50 THEN 20
40  PRINT K;" TIMES"
50  END

]RUN
50 TIMES
```

문제 35. 천만원을 년리 12.0% 의 정기예금으로 예치시켰다. 10년 후의 원리 합계를 계산하고 단리인 경우와 비교하라.

```
90  PRINT "YEAR","DANRI","BOKRI"
```

```
95 PRINT "----------------------
   -----------------"
100 P = 10000000
110 R = .12:N = 0
120 N = N + 1
130 D = P + N * P * R
140 B = P * (1 + R) ^ N
150 PRINT N,D;"     ";B
160 IF N < 10 THEN 120
165 PRINT "----------------------
   -----------------"
170 END

]RUN
YEAR          DANRI          BOKRI
-------------------------------------
1             11200000    11200000
2             12400000    12544000
3             13600000    14049280
4             14800000    15735193.6
5             16000000    17623416.8
6             17200000    19738226.9
7             18400000    22106814.1
8             19600000    24759631.8
9             20800000    27730787.6
10            22000000    31058482.1
-------------------------------------
```

> **문제36.** 첫날 1원, 다음날 2원, 그다음날 4원의 순으로 매일두배의 금액을 저금하면 천만원을 넘는것은 몇 번째 날인가를 구하는 프로그램을 작성하라.

```
10 B = - 1
20 B = B + 1
30 A = 2 ^ B
40 S = S + A
50 IF S < = 10000000 THEN 20
60 PRINT B + 1;"TH DAY"
70 END

]RUN
24TH DAY
```

> **문제37.** $Y = 5X^2 + 3X - 6$의 식에 있어서, $-10 \leqq X \leqq 10$의 범위에서 X를 0.5단위로 증가시켜 최대값과 최소값을 구하는 프로그램을 작성하라.

```
10 MAX = 0
20 MII = 99
30 X =  - 10.5
40 X = X + .5
50  IF X > 10.1 THEN 100
60 Y = 5 * X ^ 2 + 3 * X - 6
70  IF Y > MAX THEN MAX = Y
80  IF Y < MII THEN MII = Y
90  GOTO 40
100  PRINT "MAX=";MAX,"MII=";MII
110  END

]RUN
MAX=524          MII=-6.25
```

〈주의〉

여기서 최소값의 표시를 MIN으로 하는 것이 정상이나 MII대신 MIN을 사용하면 컴퓨터내의 내장 명령어와 혼합될 가능성이 있다.
예로써 행 80을 타이프하고 LIST하면

```
]LIST80

80  IF Y < M INT HENMIN = Y
```

가 된다.

> **문제38.** 두 숫자 A, B를 입력하면 숫자를 내림순으로 인쇄하는 프로그램을 작성하라.

```
10  PRINT
15  INPUT "ENTER 2 NUMBERS? ";A,B

20  IF A > B THEN  PRINT A,B: GOTO
     10
30  PRINT B,A: GOTO 10
```

```
]RUN

ENTER 2 NUMBERS? 145,678
678           145

ENTER 2 NUMBERS? -12,-15
-12           -15

ENTER 2 NUMBERS?
```

문제39. $1+2+3+\cdots\cdots+X$의 합계가 처음 200을 넘을 때, 마지막X의 값을 프린트하는 프로그램을 작성하라.

```
10 A = 0:S = 0
20 A = A + 1
30 S = S + A
40  IF S < = 200 THEN 20
50  PRINT A
60  END

]RUN
20
```

문제40. 두수 A, B의 입력에 의해 그 두수를 비교하여 A가 B에 비해 작다. 같다 혹은 크다 라는 비교 결과를 인쇄하는 프로그램을 작성하라.

```
10  INPUT "INPUT A,B? ";A,B
20  IF A < B THEN 60
30  IF A > B THEN 80
40  PRINT A;" IS EQUAL TO ";B
50  GOTO 10
60  PRINT A;" IS LESS THAN ";B
70  GOTO 10
80  PRINT A;" IS GREATER THAN ";B

90  GOTO 10

]RUN
INPUT A,B? 13,19
13 IS LESS THAN 19
INPUT A,B? 45,45
45 IS EQUAL TO 45
INPUT A,B? 78,10
78 IS GREATER THAN 10
INPUT A,B?
```

1－6 반복처리

FOR－NEXT

컴퓨터의 이점의 하나는 같은 일을 몇 번이라도 반복되게 하는 것이다. 따라서 앞에서 설명한 반복처리문 IF- GOTO문보다 더욱 간단한 반복실행문인 FOR-NEXT문에 대해 알아보자. FOR와 NEXT문은 그 사이에 있는 명령을 반복해서 처리한다.

가령 1부터 10까지의 정수를 나열하고자 한다면

```
* PROGRAM  1.23
  10  FOR N = 1 TO 10
  20  PRINT N,
  30  NEXT N
```

여기서 행번호 10에서 루프 변수의 N의 초기치가 주어지고 동시에 최종치가 기억된다. 행번호 20에서 N이 인자되고 행번호 30에서 N의 현존의 값과 이전에 주어진 최종치 패러미터가 조건을 충족시키고 있는지를 체크하고 만족하지 않을때

는 자동적으로 N의 값은 1개늘고 FOR에 계속되는 문으로 되돌아 온다.

그러면 루프 변수에는 N밖에 사용한수 없느냐하면 그렇지 않다. 변수명의 조건을 채우는 한도에서 같은 루프안에서 FOR가 붙어있는 변수와 NEXT에 붙어있는 변수가 같다면 변수라도 사용할 수 있다. 예를 들면 위의 프로그램 중에서 N을 전부 CA라는 변수로 바꾸어 놓아도 결과는 같다.

이번에는 반대로 10에서 1까지의 정수치에 대해서 이 순서대로 나열 되는 것을 생각해 보자. IF…GOTO를 사용하면

```
* PROGRAM  1.24
 ]10 I=10

 ]20 PRINT I

 ]30 I=I-1

 ]40 IF I>=1 THEN GOTO 20
```

이 된다. FOR- NEXT문을 사용하면 어떻게 될까?

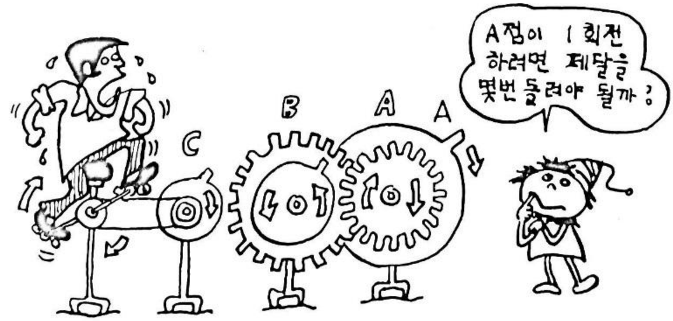
1씩 준다는 것은 ㅡ1씩 증가된다는 것과 같으므로 증분 패러미터에 음의 값을 사용한다.

```
* PROGRAM  1.25
 ]10 FOR I=10 TO 1 STEP -1

 ]20 PRINT I

 ]30 NEXT I
```

여기서 행번호10의 STEP이란 단어는 이 다음에 계속되는 수치나 식으로 루프 변수의 증가분을 나타낸다. 이 예에서 STEP- 1로 되어 있으므로 I의 값은 루프를 한번 돌 때마다 ㅡ씩 늘게된다. STEP



3중의 FOR…NEXT

이를 간단하게 도표로 그리면

행번호 FOR 변수=초기치 TO 최종치 STEP 변화폭
⋮ } 되풀이 되는 프로그램
행번호 NEXT 변수

가 된다.

그러면 여기서 루프의 네스트에 대해서 설명한다.

* PROGRAM 1.26

```
]NEW

]10 FOR X=1 TO 9

]20 FOR Y=1 TO 9

]30 PRINT X,Y

]40 NEXT Y

]50 NEXT X
```

변수Y의 루프 변수X의 루프

물론 NEXT문은 하나로 통일시킬 수 있다. 즉 행번호40을

```
]40 NEXT Y,X
```

로 하고 행번호 50은 삭제하므로서 결과는 똑같다. 그러나 주의할 것은 NEXT X, Y 로 해서는 다음 예와 같은 형태가 되어 안된다.

* PROGRAM 1.27

```
]10 FOR X=1 TO 9

]20 FOR Y=1 TO 9

]30 PRINT X,Y

]40 NEXT X

]50 NEXT Y
```

여기서 NEXT X에 맞닿는 순간에 Y루프의 변수는 완전히 없어지므로 원하는 대로의 동작을 하지 않는다. 화살표가 교차되어서는 안된다.

문제41. FOR…NEXT문을 이중으로 사용해서 다음에 표시한 도형을 그려내는 프로그램을 작성하라.

```
* * * * *
 * * * * *
  * * * * *
   * * * * *
    * * * * *
```

```
10 FOR A = 1 TO 5
20 PRINT  TAB( A);
30 FOR B = 1 TO 5
40 PRINT " *";
50 NEXT B
60 PRINT
70 NEXT A
80 END
```

문제42. 다음에 표시한 도형을 그려내는 프로그램을 작성하여라.

```
         *
        ***
       *****
      *******
     *********
    ***********
   *************
  ***************
 *****************
*******************
```

```
10 FOR I = 1 TO 10
20 PRINT  TAB( 13 - I);
30 FOR J = 1 TO 2 * I - 1
40 PRINT "*";
50 NEXT J
60 PRINT
70 NEXT I
80 END
```

문제43. 1과 30을 포함하여 두수 사이의 정수를 출력하는 프로그램을 작성하라.

```
10 FOR X = 1 TO 30
20 PRINT " ";X;
30 NEXT X

]RUN
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
 16 17 18 19 20 21 22 23 24 25 26 27
 28 29 30
```

문제44. FOR…NEXT문을 이중으로 사용해서 다음의 도형을 그려내는 프로그램을 작성하고, 또 역삼각형의 크기를 INPUT문으로 지정할 수 있도록 하라.

```
*******************
 *****************
  ***************
   *************
    ***********
     *********
      *******
       *****
        ***
         *
```

```
10 FOR I = 1 TO 10
20 PRINT  TAB( I)
30 FOR J = 2 * (11 - I) - 1 TO 1
     STEP  - 1
40 PRINT "*";
50 NEXT J
60 PRINT
70 NEXT I
80 END
```

INPUT문으로 지정할 수 있는 역삼각형

```
5  INPUT K
10 FOR I = 1 TO K
20 PRINT  TAB( I);
30 FOR J = 1 TO (2 * K + 1) - 2 *
     I
40 PRINT "*";
50 NEXT J
60 PRINT
70 NEXT I
80 END
```

```
]RUN
?7
*************
 ***********
  *********
   *******
    *****
     ***
      *
```

문제45. 같은 문자나 기호를 지정한 수 만큼 계속해서 1행에 써내는 프로그램을 활용하여 다음에 표시하는 도형을 그려내는 프로그램을 작성하라.

```
*
**
***
****
*****
****
***
**
*
```

```
10  READ K
20  FOR I = 1 TO K
30  PRINT "*";
40  NEXT I
50  PRINT
60  GOTO 10
70  DATA  1,2,3,4,5,4,3,2,1
80  END
```

문제46. 닭과 개가 모두 37마리이고 다리의 수는 102개였다.닭과 개는 각각 몇 마리인가를 구하는 프로그램을 작성하라.

```
10  FOR I = 37 TO 0 STEP  - 1
20 K = 37 - I
30  IF I * 2 + K * 4 = 102 THEN 7
     0
40  NEXT I
50  PRINT "IMPOSSIBLE"
60  END
70  PRINT "CHICKEN=";I,"DOG=";K
80  END

]RUN
CHICKEN=23      DOG=14
```

문제47. 데이타의 갯수를 입력하여 그 갯수만큼 데이타를 입력하고 평균값을 구하는 프로그램을 작성하라.

```
10  INPUT "# OF DATA=";X
20 T = 0
30  FOR I = 1 TO X
40  INPUT " DATA=";Y
50 T = T + Y
60  NEXT I
70 H = T / X
80  PRINT "AVERAGE=";H
90  END

]RUN
# OF DATA=6
 DATA=98
 DATA=86
 DATA=79
 DATA=88
 DATA=85
 DATA=73
AVERAGE=84.8333334
```

문제48. 4194304는 2의 몇 제곱인가를 구하는 프로그램을 작성하라.

```
10 X = 4194304
20  FOR I = 1 TO 30
30 X = X / 2
40  IF X = 1 THEN 60
50  NEXT I
60  PRINT "4194304 IS 2 EXPONENTI
     AL ";I
70  END
]RUN
4194304 IS 2 EXPONENTIAL 22
```

문제49. 5만원을 복리 7.8%로정기예금할 때 원리합계가 18만원을 넘는 것은 몇년후인가를 구하는 프로그램을 작성하라. 단,
원리합계
=원금×(1+연이율)^{년수}

```
10  FOR I = 1 TO 50
20 X = 50000 * (1 + .078) ^ I
30  IF X > 180000 THEN  PRINT I;"
      TH YEAR": GOTO 50
40  NEXT I
50  END

]RUN
18 TH YEAR
```

1-7 서브루틴(Subroutine)

GOSUB

만약 당신이 만들려고 하는 프로그램이 몇개의 다른 장소에서 같은 처리를 행한다면 같은 것을 몇번이고 쓰는 것은 어리석은 일이다. 따라서 이런것을 배제하는 것이 서브루틴(subroutine)의 사고방식이다.

따라서 GOTO Subprogram인 GOSUB가 실행되면 프로그램은 GOSUB의 바로 뒤에 주어지는 행번호로 분기한다. 그런데 무조건 GOTO와 달라서 GOSUB는 그 GOSUB가 있던 장소를 기억하고 있고 서브루틴 내에서 RETURN을 발견하면 GOSUB가 있던 바로 다음으로 되돌아 온다.

IF-GOSUB

다음 프로그램을 검토해 보자.

* PROGRAM 1.28

```
]10 INPUT X
]20 IF X>0 THEN GOSUB 100
]30 IF X<0 THEN GOSUB 200
]40 IF X=0 THEN GOSUB 300
]50 END
]100 PRINT "PLUS"
]110 RETURN
]200 PRINT "MINUS"
]210 RETURN
]300 PRINT "ZERO"
]310 RETURN
```

이 프로그램은 수 X의 입력을 요구하고 플러스 일때는 "PLUS", 마이너스일때는 "MINUS", 0일때에는 "ZERO"라 인쇄하는 것으로 행번호 100~110, 200~210, 300~310은 독립된 1개의 프로그램이며 어느것이든지 불러내어 사용할 수 있는데 이것을 부프로그램(subprogram)이라 한다.

GOSUB
-RETURN

항상 부 프로그램의 끝은 RETURN이며, RETURN으로 되돌아 가는 장소는 불러냈던 GOSUB문 바로 다음이다.

만약 수 X를 0으로 입력하였다면 행번호20과 30의 조건에 맞지 않으므로 뛰고

행번호 40에서 GOSUB300이 되어 행번호300에 의해 ZERO라고 프린트하고 RETURN를 만나 GOSUB를 불러냈던 장소인 행번호40바로 다음인 행번호50에서 끝내게 된다.

ON-GOSUB 또 하나의 예를 들어 보자.

양의 정수 X를 입력하고 이 값이 1일때에는 직각삼각형의 면적을, 2일때에는 정사각형의 면적을 구하는 프로그램이다.

* PROGRAM 1.29

```
]NEW

]10 INPUT X

]20 ON X GOSUB 100,200

]30 END

]100 INPUT "L=";L

]110 INPUT "H=";H

]120 S=L*H/2

]130 PRINT "BASE=";L;" HEIGHT=";H

]140 PRINT " AREA OF TRIANGLE=";S

]150 RETURN

]200 INPUT "SIDE=";L

]210 S=L*L

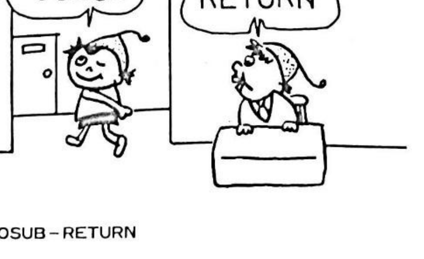
]220 PRINT "SIDE=";L

]230 PRINT " AREA OF SQUARE=";S

]240 RETURN

]RUN
?1
L=3
H=5
BASE=3 HEIGHT=5
 AREA OF TRIANGLE=7.5

]RUN
?2
SIDE=4
SIDE=4
 AREA OF SQUARE=16
```



GOSUB-RETURN



ON-GOTO

행번호20의 ON X GOSUB 100, 200에서는 X가 1이면 행번호100, X가 2일 때에는 행번호200의 부프로그램으로 분기한다. X가 1혹은 2가 아닐때에는 행번호30으로 진행하여 끝나게 된다. ON X GOSUB는 ON X GOTO와 같은 작용을 하나 GOSUB는 GOSUB문의 바로 다음 번호로 되돌아 오지만 GOTO에서는 되돌아 오지 않는다.

POP

계층구조를 지닌 서브루틴에 대해 한 단계를 뛰어넘어 RETURN되도록 POP명령을 사용할 수 있다. 즉 POP명령은 다른곳으로 실제로는 분기하지 않는 RETURN 역할을 한다. 이 명령 다음에 RETURN을 만나면 그 루틴을 부른 가장 최근 실행된 부프로그램이 끝나고 돌아갈 GOSUB다음문으로 접속되지 않고 두 번째로 최근에 실행된 GOSUB문 다음으로 분기된다.

POP명령이 실행되면 실제 컴퓨터에서는 RETURN번지들이 들어있는 스택의 가장 위에 있는 번지를 소거해 버리므로 POP라고 한다.

POP가 없는 다음 프로그램을 보자.

```
* PROGRAM  1.30
  10  REM  RETURN TESTING
  20  REM  MAIN ROUTINE
  30  GOSUB 100
  40  END
  100  REM  SUB 1ST
  110  GOSUB 200
  120  RETURN
  200  REM  SUB 2ND
  220  RETURN

  ]TRACE

  ]RUN
  #10 #20 #30 #100 #110 #200 #220 #120 #40
```

행번호220다음 120을 거쳐 40으로 되돌아가지만 다음의 POP명령이 있는 프로그램의 예에서는 120을 거치지 않고 바로 행번호 40으로 접속되고 있다. 즉,

MAIN→SUB 1st→SUB 2nd→SUB 1st→MAIN

으로 되어야 할 것이 POP명령을 써서

MAIN→SUB 1st→SUB 2nd→MAIN

으로 되고 있다.

다음 프로그램은 POP명령을 사용한 것으로 TRACE명령을 이용해 추적시켜 보았다.

```
* PROGRAM  1.31
  10  REM  RETURN TESTING
  20  REM  MAIN ROUTINE
  30  GOSUB 100
```

```
40  END
100  REM  SUB 1ST
110  GOSUB 200
120  RETURN
200  REM  SUB 2ND
210  POP
220  RETURN

]TRACE

]RUN
#10 #20 #30 #100 #110 #200 #210 #220 #40
```

여기서 행번호110의 GOSUB 200의 바로 다음 행번호120 RETURN문으로 돌아가지 않고 행번호30 GOSUB 100의 바로 다음 행번호40으로 RETURN되었음을 알수있다.

문제50. 다음과 같은 결과가 나오도록 하는 프로그램을 작성하라. 단 브랭크(blank)행과 그것에 이어진 행은 서브루틴에서 인쇄하도록 한다.

```
]RUN

PART #1
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
PART #2
*               *               *

PART #3
$/$/$/$/$/$/$/$/$/$/$/$/$/$/$/$/$/$/$/$/
```

```
10  GOSUB 200
20  FOR J = 1 TO 40
30  PRINT "&";
40  NEXT J
50  GOSUB 200
60  FOR J = 1 TO 3
70  PRINT "*",
80  NEXT J
90  GOSUB 200
100  FOR J = 1 TO 20
110  PRINT "$/";
120  NEXT J
130  END
200  PRINT
210 I = I + 1
220  PRINT "PART #";I
230  RETURN
```

문제51. 다음 프로그램이 실행되는 순서대로 행번호를 나열하라.

```
10  READ X
20  IF X = 0 THEN 100
30  GOSUB 60
40 T = T + X
50  GOTO 10
60  IF X < 0 THEN 80
70 P = P + X
80  RETURN
90  DATA 5,2,-9,0,3,5
100  PRINT "T= ";T,"P= ";P

]RUN
T= -2           P= 7
```

```
]TRACE

]RUN
#10 #20 #30 #60 #70 #80 #40 #50 #10
#20 #30 #60 #70 #80 #40 #50 #10 #20
#30 #60 #80 #40 #50 #10 #20 #100
```

문제52. 수X를 입력하여 $(3X^2-3)+(3X^2-3)^2-6X$를 구하는 프로그램을 작성하라.

```
10  INPUT "X? ";X
20  GOSUB 100
30 ANS% = A + A ^ 2 - 6 * X
40  PRINT "X= ";X
50  PRINT "(3X^2-3)+(3X^2-3)^2-6X
     = ";ANS%
60  END
100 A = 3 * X ^ 2 - 3
110  RETURN

]RUN
X? 7
X= 7
(3X^2-3)+(3X^2-3)^2-6X= 20838
```

문제53. 수X를 입력하여, X=1 일때 「HELLO」, X=2 일 때 「GOOD MORNING」, X=3 일 때 「GOOD AFTERNOON」으로 인쇄하는 프로그램을 작성하다.

```
10  INPUT "X? ";X
20  ON X GOSUB 100,200,300
30  END
100  PRINT "HELLOW"
110  RETURN
200  PRINT "GOOD MORNING"
210  RETURN
300  PRINT "GOOD AFTERNOON"
310  RETURN

]RUN
X? 1
HELLOW

]RUN
X? 2
GOOD MORNING

]RUN
X? 3
GOOD AFTERNOON
```

문제54. 수X를 입력하여 X=1 일때 「I AM A BOY.」, X=2 일때 「YOU ARE A BOY.」, X=3 일때 「WE ARE BOYS.」라고 인쇄하는 프로그램을 작성하라.

```
10  INPUT X
20  ON X GOSUB 100,200,300
30  END
100  PRINT " I AM ";
105  GOSUB 400
110  RETURN
200  PRINT " YOU ARE ";
205  GOSUB 400
210  RETURN
300  PRINT " WE ARE BOYS."
310  RETURN
400  PRINT "A BOY."
410  RETURN

]RUN
?1
 I AM A BOY.

]RUN
?2
 YOU ARE A BOY.

]RUN
?3
 WE ARE BOYS.
```

1 — 8 표준함수 (Library Function)

수학적으로 유용한 많은 연산 즉, θ 의 sine이라든가, X의 평방근이라든가 하는 계산이 프로그램의 가운데에서 실행될 수 있다. 컴퓨터에서는 많은 **표준함수**(標準函數 : library function)가 내장되어 있으며, DEF문에 의해 독자적인 내부의 루틴을 만드는 사용자 함수도 있다.

```
SQR (X)
```

SQR

SQR(X)는 수치 X의 제곱근을 구한다.

```
* PROGRAM 1.32
]10 INPUT X

]20 PRINT SQR(X)
```

연산식 SQR(X)에서 SQR은 함수명이고 X는 인수라고 불린다. SQR(X)함수에서의 인수(argument)는 BASIC의 어떠한 연산식도 가능하며 음수값만 가지지 않으면 된다. 예로써

```
* PROGRAM 1.33
]10 INPUT X

]20 PRINT SQR(3*X+5)

]RUN
?25
8.94427191
```

SQR(3 * X + 5)는 연산식 3 * X + 5가 음수값만 안나오면 규칙에 맞는 형식으로 받아진다.

함수에서 인수는 꼭 괄호로 묶어져야 한다. 수학적 표준함수를 계산할 때 컴퓨터는 우선적으로 인수를 계산한 다음 그 결과를 가지고 함수의 값을 계산한다.

어떤 경우에는 어느 변수가 양수인지 음수인지를 알 필요가 있다.

SGN

SGN(X)함수는 그 결과가 3가지의 값 즉, 1, 0, −1만 갖는다.

X 〉0 이면 1　　(예)SGN(0.4) = 1
X = 0 이면 0　　(예)SGN(0) = 0
X 〈0 이면 −1　　(예)SGN(− 5) = −1

다음 예의 프로그램을 타이프 해 보자.

```
* PROGRAM 1.34
10 FOR X = 20 TO - 20 STEP - 5

20 A = SGN (X)
```

```
30  PRINT A;"  ";
40  NEXT X

]RUN
1  1  1  1  0  -1  -1  -1  -1
]
```

이것은 20부터 증가치를 −5로 하여 −20까지의 숫자의 SGN을 구한 것이다.

ABS(X)함수는 어느 숫자의 절대값을 구하는데 사용될 수 있다. ABS

ABS(X)함수는 다음 법칙에 따라 값을 갖는다.

X〉= 0 일때 ABS(X)=X

X〈 0 일때 ABS(X) =−1 * X

앞서의 프로그램중

```
20 A =  SGN (X)
```

대신에

```
]20 A=ABS(X)
```

로 하고 RUN시켜보면 다음과 같이 될 것이다.

```
]RUN
20  15  10  5  0  5  10  15  20
]
```

다음과 같은 짧은 프로그램을 한번 실행해 보자. RND

```
* PROGRAM  1.35
]NEW

]100 PRINT RND(1)

]110 GOTO 100
```



RND

행번호100의 RND는 Random이라는 것이다. RND는 난수를 발생한다. RUN 시켜보자. 계속 난수를 발생시킬 것이다.

CTRL

C

로 멈추어 보자.

```
]RUN
.973136996
.103117626
.0177148333
.779343355
.551834438
.617419111

BREAK IN 100
]
```

0과 1사이의 난수가 발생되어 있을 것이다. 물론 여러분이 실행시켰을 때는 여기의 예와는 다를 것이다. 컴퓨터 자체의 난수발생이니까. 어쨌던 여기서RND는 0.0000000001 부터 0.9999999999까지의 수를 발생시킨다.

다음에 행번호 100을 다음과 같이 바꾸어 보자.

```
* PROGRAM 1.36
]100 PRINT RND(6)

]RUN
.960296981
.547150891
.802192734
.814107273
.131137465

BREAK IN 100
]
```

한번 더 CTRL-C 로 멈추어 보면, 마찬가지로 0과 1사이의 난수이다. RND(X)는 인수X가 양의 정수이면 항상0과 1사이의 값을 갖는 난수를 만든다.

INT

0과 1사이의 난수에서는 어딘지 부족하다. 조금 더 범위를 넓혀 0부터 9까지의 정수의 난수를 얻도록 하자.

```
* PROGRAM 1.37
]NEW

]100 X=RND(1)

]110 X=X*10
```

```
]120 X=INT(X)

]130 PRINT X
```

행번호 120의 INT(X)는 인수X의 값보다 작은 정수를 준다. 즉 X>0일 때는 소숫점 이하를 끊어낸 정수로, X<0는 소숫점 이하를 내린 정수로 된다. 예를 들면 X=4.17382이면 INT(X)는 4가되고, X=−2.7268이면 INT(X)는 −3이 된다. INT뒤의 괄호속에는 수식이나 변수를 포함할 수 있다.

프로그램을 RUN시켜 보자. 0부터 9까지가 아니고 1부터 10까지의 난수로 바꾸려면 다음행을 첨가하면 된다.

```
]125 X=X+1
```

프로그램을 RUN시키고 어떻게 되는지 보자.

이 프로그램은 다음과 같이 1행으로 처리할 수도 있다.

```
]100 PRINT INT(10*RND(1))+1
```

난수의 사용법을 알았으므로 1쌍의 주사위를 만들어 보자.

```
* PROGRAM  1.38
]NEW

]100 PRINT "WHITE DICE"

]110 PRINT INT(6*RND(1))+1

]120 PRINT "RED DICE"

]130 PRINT INT(6*RND(1))+1
```

이 프로그램은 "WHITE DICE", "RED DICE"의 각각의 주사위로 1부터 6까지의 난수를 발생시킨다. 한번 더 주사위를 던지려면 RUN을 시키면 된다.

BASIC에서는 여러가지 삼각함수의 값을 구할 수 있다.

SIN(X), COS(X), TAN(X)는 각 인수 X의 sine, cosine, tangent의 값을 나타낸다. 여기서 X는 라디언(radian) 값이어야 한다. 각도를 라디언으로 바꾸려면 각도에 π를 곱한 후 180으로 나누면 된다.

SIN
COS
TAN

라디언=각도 $* \frac{\pi}{180}$ 단, $\pi=3.14159265\cdots$

다음은 삼각함수에 관한 프로그램이다.

```
* PROGRAM  1.39
10  INPUT "VALUE?  ";A
20 X = A * 3.1415926 / 180
30  PRINT A;" DEGREES= ";X;" RADI
     ANS"
40  PRINT "SIN(";A;")="; SIN (X)
```

```
50  PRINT "COS(";A;")="; COS (X)
60  PRINT "TAN(";A;")="; TAN (X)
70  PRINT
80  GOTO 10

]RUN
VALUE?  30
30 DEGREES= .523598767 RADIANS
SIN(30)=.499999992
COS(30)=.866025408
TAN(30)=.577350257
```

ATN

그리고 BASIC에서 사용되는 역삼각함수는 arctangent 함수ATN(X)뿐이다. ATN(X) 함수의 결과치는 라디언으로 나온다. 그러므로 1의 arctangent는 $\frac{\pi}{4}$ radian=0.785398103이다. ATN(X) 함수는 수학에서의 arctangent함수와 마찬가지로 $-\frac{\pi}{2}$와 $\frac{\pi}{2}$라디언 사이의 값으로 주어진다.

기타 역 삼각함수는 이상의 함수를 이용하여 만들 수 있다.

arcsine을 구하려면 다음과 같은 삼각함수 공식을 이용하여야 할 것이다.

$$\text{arcsine}(X) = \text{arctangent}\ \frac{X}{\sqrt{1-X^2}}$$

그러므로 어느 각도의 sine값이 X인 각도를 구하려면

연산식 ATN(X/SQR(1－X∧2))가 사용된다. 이 각도는 라디언으로 얻어지며 $-\frac{\pi}{2}$와 $\frac{\pi}{2}$의 범위일 것이다. 그러나 X는 －1과 1을 포함치 않는 사이의 숫자여야 한다.

X의 arccosine을 구하려면 연산식 ATN(SQR(1－X∧2)/X) 를 사용한다. 이것은 $-\frac{\pi}{2}$와 $\frac{\pi}{2}$ 사이의 각도로 주어진다. 이 연산식에서 X는 －1과 1사이의 숫자로서 0는 아니다.

다음 프로그램은 X의 arcsine과 arccosine을 라디언과 각도로 구한 것이다.

```
* PROGRAM  1.40
 10  INPUT X
 20 S =  ATN (X /  SQR (1 - X ^ 2))
 30 C =  ATN ( SQR (1 - X ^ 2) / X)
 40  PRINT "THE ANGLE WHOSE SINE IS ";X;" IS ";S;" RADIANS"
 50  PRINT "THE ANGLE WHOSE COSINE IS ";X;" IS ";C;" RADIANS"
 60  PRINT "THE ANGLE WHOSE SINE IS ";X;" IS ";S * 180 / 3.141
     59265;" DEGREES"
 70  PRINT "THE ANGLE WHOSE COSINE IS ";X;" IS ";C * 180 / 3.1
     4159265;" DEGREES"
 80  GOTO 10

 ]RUN
 ?0.5
 THE ANGLE WHOSE SINE IS .5 IS .523598776 RADIANS
```

```
THE ANGLE WHOSE COSINE IS .5 IS 1.04719755 RADIANS
THE ANGLE WHOSE SINE IS .5 IS 30 DEGREES
THE ANGLE WHOSE COSINE IS .5 IS 60.0000001 DEGREES
?
```

LOG

LOG함수는 밑수(base)를 e로 하는 자연대수를 구하는데 사용된다. X의 자연대수를 구하려면 LOG(X)를 타이프한다. 여기서 밑수를 10으로 하는 상용대수와 혼동해서는 안된다.

상용대수는 다음 식을 사용해서 자연대수로부터 구할 수 있다.

$$\log_{10}(X) = \frac{\ln(X)}{\ln(10)}$$

여기서 ln(X)는 자연대수를 나타낸다. 그러므로 $\log_{10}(X)$를 구하기 위해서는 간단히 LOG(X) / LOG(10)을 타이프하면 된다. LOG함수의 인수는 항상 양수이어야 한다.

다음은 상용대수를 구하는 프로그램이다.

```
* PROGRAM  1.41
 ]10 INPUT X

 ]20 T=LOG(X)/LOG(10)

 ]30 PRINT "THE COMMON LOG OF ";X;" IS ";T

 ]40 GOTO 10

]RUN
?1000
THE COMMON LOG OF 1000 IS 3
?0.0001
THE COMMON LOG OF 1E-04 IS -4
?
```

EXP

EXP함수는 지수함수 e^x값을 구하는데 사용된다. 이 숫자는 자연대수의 밑수와 같다. EXP는 exponential(지수)를 의미하며 e^x를 구하려면 EXP(X)를 타이프 한다.

즉 e^5은 EXP(5)가 된다.

```
* PROGRAM  1.42
]PRINT EXP(5)
148.413159

]PRINT EXP(1)
2.71828183
```

따라서 e는 2.71828183의 값을 갖는다.

DEF FN

지금까지 함수를 보아 왔지만 여기서는 이들 함수를 조합시킨 새로운 함수로서

정의하는 DEF FN 명령을 설명하고자 한다.

DEF는 「정의한다(define)」라는 뜻이며 여기에 「함수(function)」의 FN을 붙여 DEF FN 명령으로 사용하여 다른 함수를 정의할 수 있다. DEF의 중요한 장점은 어느 함수가 프로그램내의 여러 장소에서 연산되어 지지만 그 필요한 함수의 연산식을 한번만 기술하여도 된다는 점이다. DEF문의 형식은

| DEF FN 함수명(변수명)＝연산식 |

함수명은 A, F, G3등의 숫자용 변수명에 적합한 것이면 된다. 함수명 뒤에 나오는 변수 즉 인수는 항상 괄호내에 있어야 하며 적당한 숫자용 변수라도 된다.

예로서 DEF FN A(X)＝2 ∗ X ∧ 2 + 3 ∗ X + 1은 $2X^2+3X+1$을 FN A(X)로 정의하며 DEF FN B(X)＝SIN(X)∧2+COS(X)∧2 는 $\sin^2 X+\cos^2 X$을 FN B(X)로 정의한다. FN B(X)은 항상 1로 된다. 여기서 A나 B는 함수명, X는 변수명, 그리고 2 ∗ X ∧ 2 + 3 ∗ X + 1등은 함수의 값을 연산하는데 사용되는 연산식이다. 이때 X＝1이면 2 ∗ 1 ∧ 2 + 3 ∗ 1 + 1 = 6 이므로 FN A(X)＝6 이 된다.

DEF FN을 이용해 운동에너지에 관한 프로그램을 작성해 보자.

```
* PROGRAM  1.43

]NEW

]10 DEF FN E(V)=1/2*M*V^2

]20 M=5.5:V=3.5

]30 PRINT FN E(V),FN E(V*2),FN E(V*3)

]RUN
33.6875         134.75          303.1875
```

여기서 행번호 30의 3개 변수는 각각 다르다. 초속이 V일때의 에너지, 속도가 2배, 3배로 커졌을 때의 운동에너지가 표시된다. 즉 괄호안의 변수는 DEF문에서 사용되는 변수와 같아도 되고 달라도 상관없다. 단지 여기서 DEF문을 사용하지 않는다면 행번호 30을 실행시키기 위해 행번호 10의 연산식이 두번 더 나타나야 한다.

그러므로 분량이 많은 긴 프로그램에서 몇 번인가 같은 형의 함수를 사용할 때 DEF FN은 매우 편리하게 이용된다.



참 고 　　표준함수 (標準函數 : library function)

함　수	BASIC기호	의　미	예
절 대 치	ABS (X)	X의 절대치 | X |	ABS(−3)=3 ABS(2)=2
부　호	SGN (X)	X>0 이면 1 X=0 이면 0 X<0 이면 −1	SGN(0.4)=1 SGN(0)=0 SGN(−5)=−1
정 수 화	INT (X)	X를 넘지 않는 최대의 정수	INT(3.8)=3 INT(0.4)=0 INT(−3.2)=−4
평 방 근	SQR (X)	$\sqrt{X}$ 단, $X \geq 0$	SQR(9)=3 SQR(0)=0
난　수	RND (X)		난수를 만든다
삼각함수 사　인	SIN (X)	sin x	이 경우 각도는 래디언 (radian)으로 표현한다. ※ 래디언과 각도와의 관계 래디언＝각도 $\times \frac{\pi}{180}$
삼각함수 코 사 인	COS (X)	cos x	
삼각함수 탄 젠 트	TAN (X)	tan x	
삼각함수 아크탄젠트	ATN (X)	$\tan^{-1} x$	
지수함수	EXP (X)	e^x (e=2.7182818)	EXP (1)=2.7182818 EXP (0)=1
자연대수	LOG (X)	$\log_e X$ 단, $X>0$	LOG (3)=1.0986123 ※ 상용대수를 구하는 방법 상용대수＝$\frac{\text{LOG (X)}}{\text{LOG (10)}}$

문제 55. 난수를 2개 만들어 그차의 절대값을 구하는 프로그램을 작성하라.

```
10 A =  RND (1)
20 B =  RND (1)
30 X =  ABS (A - B)
40  PRINT A;" AND ";B
50  PRINT "DIFFERENCE=";X
60  END

]RUN
.0177148333 AND .779343355
DIFFERENCE=.761628522
```

문제56. $\log_e 10.2$를 구하는 프로그램을 작성하라.

```
10 A =  LOG (10.2)
20 PRINT A
30 END

]RUN
2.32238772
```

문제57. 수X 와Y 를 입력하여 X+Y의 값이 플러스이면 1, 마이너스 이면 −1, 0 이면 0 를 프린트하는 프로그램을 작성하라.

```
10  INPUT "X=";X
15  INPUT "Y=";Y
20  PRINT "ANSWER=("; SGN (X + Y)
    ;")"
30  END

]RUN
X=2
Y=5
ANSWER=(1)

]RUN
X=5
Y=-8
ANSWER=(-1)

]RUN
X=-7
Y=7
ANSWER=(0)
```

문제58. 난수를 2 개 만들어 그 2 개의 가감승제의 결과를 구하는 프로그램을 작성하라.

```
10 A =  RND (1)
20 B =  RND (1)
30  PRINT "A=";A,"B=";B
40  PRINT "A+B=";A + B,"A-B=";A -
     B,"A*B=";A * B,"A/B=";A / B
50  END
60  END

]RUN
A=.960296981      B=.547150891
A+B=1.50744787  A-B=.41314609
A*B=.525427349  A/B=1.75508621
```

문제59. 아래와 같이 1 과50를 포함한 두 수 사이의 난수를 알아내는 게임을 하는 사람이 5번만 질문하도록 하는 프로그램을 작성하라.

```
]RUN
WHAT IS YOUR GUESS? 36
TOO HIGH
WHAT IS YOUR GUESS? 28
TOO HIGH
WHAT IS YOUR GUESS? 15
TOO HIGH
WHAT IS YOUR GUESS? 10
TOO LOW
WHAT IS YOUR GUESS? 13
YOU ARE CORRECT!

]RUN
WHAT IS YOUR GUESS? 10
TOO HIGH
WHAT IS YOUR GUESS? 1
TOO LOW
WHAT IS YOUR GUESS? 2
TOO LOW
WHAT IS YOUR GUESS? 3
TOO LOW
WHAT IS YOUR GUESS? 9
TOO HIGH
 YOU HAVE MADE TOO MANY MISTAKES.
THE NUMBER WAS 7
```

```
10 R =  INT (50 *  RND (1) + 1)
20  FOR X = 1 TO 5
30  INPUT "WHAT IS YOUR GUESS? ";
     G
40  IF G = R THEN 120
50  IF G < R THEN 70
60  PRINT "TOO HIGH": GOTO 80
70  PRINT "TOO LOW"
```

```
80 NEXT X
90 PRINT " YOU HAVE MADE TOO MAN
    Y MISTAKES."
100 PRINT "THE NUMBER WAS ";R
110 END
120 PRINT "YOU ARE CORRECT!"
```

문제60. 0과 9를 포함한 두수 사이에서 50개의 난수를 고르고 0~4까지 몇개 5~9까지 몇 개가 각각 포함되는지를 나타내는 프로그램을 작성하라.

```
10 FOR X = 1 TO 50
20 R = INT (10 * RND (1))
30 IF R < 5 THEN A = A + 1: GOTO
    50
40 B = B + 1
50 NEXT X
60 PRINT " (0-4): ";A
70 PRINT " (5-9): ";B

]RUN
 (0-4): 28
 (5-9): 22
```

문제61. 다음과 같이 각 수를 입력시켜 나눗셈을 행하고 몫과 나머지를 인쇄하는 프로그램을 작성하라.

```
]RUN
A DIVIDED BY B IS? (A,B) 45,57

45/57=0...45
```

```
10 INPUT "A DIVIDED BY B IS? (A,
    B) ";A,B
15 PRINT
20 S = SGN (A)
30 A = ABS (A):B = ABS (B)
40 PRINT A;"/";B;"="; INT (A / B
    );"...";
50 PRINT INT ((A / B - INT (A /
    B)) * B + .5) * S
```

```
]RUN
A DIVIDED BY B IS? (A,B) 25,4

25/4=6...1

]RUN
A DIVIDED BY B IS? (A,B) 32,8

32/8=4...0
```

문제62. 951424를 읽고 위2자리와 아래 2자리를 인쇄하는 프로그램을 작성하라.

```
10 READ A
20 B = INT (A / 10000)
30 C = INT (A / 100)
40 D = A - 100 * C
50 PRINT B,D
60 END
70 DATA 951424

]RUN
95              24
```

문제63. 250과 275를 포함한 두수 사이의 난수를 12개 만드는 프로그램을 작성하라.

```
10 FOR X = 1 TO 12
20 R = INT (26 * RND (1) + 250)

30 PRINT R;" ";
40 NEXT X

]RUN
257 271 253 271 264 273 261 250 253 275 253 275
]RUN
268 275 270 257 268 269 252 272 269 251 256 271
```

문제64. 어떤 수 X를 읽고, 그 수의 절대값을 인쇄하는 프로그램을 작성하라.

```
10  INPUT X
20  PRINT "X=";X;" ABSOLUTE VALUE
     =";  ABS (X)
30  END

]RUN
?-134
X=-134 ABSOLUTE VALUE=134
```

문제65. 직각 삼각형의 두 변의길이 A, B를 읽고 빗변의 길이 C를 구하는 프로그램을 작성하라. 단, $C=\sqrt{A^2+B^2}$ 이다.

```
10  READ A,B
20 C =  SQR (A ^ 2 + B ^ 2)
30  PRINT "C=";C
40  DATA  6,8
50  END

]RUN
C=10
```

문제66. 데이타 16, 144, 18.6을 읽고 각각의 제곱근을 구하는프로그램을 작성하라.

```
10  READ A,B,C
20  PRINT  SQR (A), SQR (B), SQR
     (C)
30  DATA  16,144,18.6
40  END

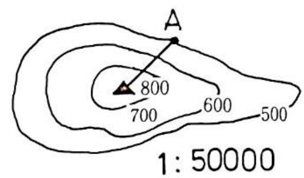
]RUN
4              12              4.31277173
```

문제67. 마이너스 온도를 2개 입력하여 그 차를 구하는 프로그램을 작성하라.

```
10  INPUT "X,Y>>";X,Y
20  PRINT "X=";X,"Y=";Y
30  PRINT "DIFFERENCE="; ABS (X - Y)
40  END

]RUN
X,Y>>-23,-32
X=-23              Y=-32
DIFFERENCE=9
```

문제68. 아래의 지도에서 A지점과 정상간의 거리는 28mm였다. 정상은 표고 800m 이다. A지점과 정상간의 직선거리와 수평거리를 구하는 프로그램을 작성하라.




```
10  READ A
20 B = 50000 * 28 / 1000
30 C =  SQR (800 ^ 2 + B ^ 2)
40  PRINT "LINEAL DISTANCE=";C;"
     M"
50  PRINT "HORIZONTAL DISTANCE=";
     B;" M"
60  DATA  28
70  END

]RUN
LINEAL DISTANCE=1612.45155 M
HORIZONTAL DISTANCE=1400 M
```

문제69. 1부터 10까지의 정수에 대한 제곱근을 프린트하는 프로그램을 작성하라.

```
10  FOR I = 1 TO 10
20  PRINT "SQR(";I;") ="; SQR (I)

30  NEXT I
40  END

]RUN
SQR(1) =1
SQR(2) =1.41421356
```

```
SQR(3) =1.73205081
SQR(4) =2
SQR(5) =2.23606798
SQR(6) =2.44948974
SQR(7) =2.64575131
SQR(8) =2.82842713
SQR(9) =3
SQR(10) =3.16227766
```

문제70. 소숫점이하 2자리의 수 X를 입력하여 반올림하고 소숫점 첫째 자리까지를 구하는 프로그램을 작성하라.

```
10  INPUT "X=";X
20 Y =  INT (X * 10 + .5) / 10
30  PRINT "X=";X;" ROUND OFF =";Y

40  END

]RUN
X=24.25
X=24.25 ROUND OFF =24.3
```

문제71. 수X를 입력하고 그 제곱근을 소숫점 3자리까지 구하는 프로그램을 작성하라.

```
10  INPUT "X=";X
20 X1 =  SQR (X)
25 X1 =  INT (X1 * 1000) / 1000
30  PRINT "X=";X;" SQR(";X;")=";X
    1
40  END

]RUN
X=16.27
X=16.27 SQR(16.27)=4.033
```

문제72. 열차가 언덕을 오를때100 m 간의 거리에 대해 5 m 상승하고 있다. 이 언덕의 경사각을 구하는 프로그램을 작성하라. 단, 단위는 도(degree)로 표시하라.

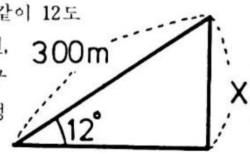
```
10 A = 5 / 100
20 X =  ATN (A)
30 ANS = X / 3.14159 * 180
40  PRINT ANS;" DEGREE"
50  END

]RUN
2.86240765 DEGREE
```

〈참고〉 $\tan\theta = \frac{5}{100}$ 에서 $\theta = \tan^{-1}\frac{5}{100}$ 로 된다.

ATN은 arctangent의 약자로 사용했다.

문제73. 오른쪽 그림과 같이 12도의 언덕을 300m 걸으면, 높이는 몇 m 인가를 구하는 프로그램을 작성하라.




```
10 A = 12 / 180 * 3.14159
20 X =  SIN (A) * 300
30  PRINT X;" M"
40  END

]RUN
62.3734554 M
```

문제74. sin (A+B) = sin (A) cos (B) + cos(A) sin(B)의 좌변과 우변을 각기 달리 계산하여 인쇄하는 프로그램을 작성하라. 단 A는 15도, B는 30도이다.

```
10 A = 15 / 180 * 3.14159
20 B = 30 / 180 * 3.14159
30  PRINT "SIN(15+30)="; SIN (A +
    B)
40  PRINT "SIN(15)COS(30)+COS(15)
     SIN(30)="; SIN (A) *  COS (B
     ) +  COS (A) *  SIN (B)
50  END

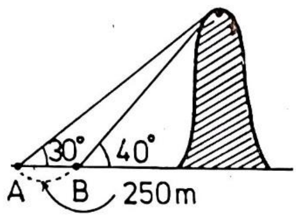
]RUN
SIN(15+30)=.707106312
SIN(15)COS(30)+COS(15)SIN(30)=.707106293
```

문제75. sin 32° 와 cos32° 의 차를 구하는 프로그램을 작성하라.

```
10  READ A
20 B = A / 180 * 3.14159
30 C =  SIN (B) -  COS (B)
40  PRINT  ABS (C)
50  DATA  32
60  END

]RUN
.318129482
```

문제76. 아래의 그림과 같이 A, B 점에서 산꼭대기를 바라보는 각은 각각 30°, 40°였다. A, B간을 250m 라 할때 산의 높이를 구하는 프로그램을 작성하라.




```
10  READ A,B,C
20 A1 = A / 180 * 3.14159
30 B1 = B / 180 * 3.14159
40 C1 = (B - A) / 180 * 3.14159
50 H = C *  SIN (A1) *  SIN (B1) /
     SIN (C1)
60  PRINT "HIGHT=";H;" M"
70  DATA  30,40,250
80  END

]RUN
HIGHT=462.707979 M
```

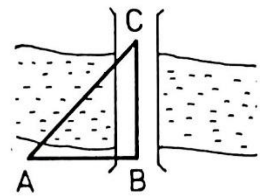
문제77. X도를 입력하여 sin (X) + $\sqrt{3}$ cos (X)를 구하는 프로그램을 작성하라

```
10  INPUT X
20 X1 = X / 180 * 3.14159
30 ANS =  SIN (X1) +  SQR (3) *  COS
    (X1)
40  PRINT "SIN(";X;")+SQR(3)*COS(
    ";X;")=";ANS
50  END

]RUN
?45
SIN(45)+SQR(3)*COS(45)=1.931852
```

문제78. 다음의 그림과 같이 강에 다리가 놓여 있다. 삼각형 ABC 가 직각 삼각형일 때, AB=15 m, BC=18m 였다면 각 BAC는 몇도인가를 구하는 프로그램을 작성하라.

〈참고〉 $\tan\theta=\frac{18}{15}$




```
10 X = 18 / 15
20 A =  ATN (X)
30 ANS = A / 3.14159 * 180
40  PRINT ANS;" DEGREE"
50  END

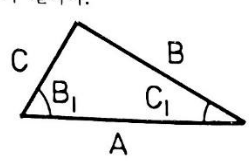
]RUN
50.1944713 DEGREE
```

문제79. 각 30°의 cosec를 구하는 프로그램을 작성하라.
단, $\text{cosec } 30°=\frac{1}{\sin 30°}$ 이다.

```
10 A = 30 / 180 * 3.14159
20 X =  SIN (A)
30  PRINT "COSEC(30)=";1 /  SIN (
     A)
40  END

]RUN
COSEC(30)=2.00000161
```

문제80. 아래의 삼각형에서 A 의 길이를 구하는 프로그램을 작성하라.
단, $A=C\cos B_1°+B\cos C_1°$ 이며 $B=15$, $C=10$, $B_1=45$, $C_1=20$ 이다. 여기서 A, B, C는 변의 길이다.




```
10 B1 = 45 / 180 * 3.14159
20 C1 = 20 / 180 * 3.14159
30 A = 10 *  COS (B1) + 15 *  COS (C1)
```

```
40 PRINT A
50 END

]RUN
21.1664633
```

문제 81. -0.1263의 $\tan^{-1}$을 구하는 프로그램을 작성하라.

```
10 PRINT ATN ( - .1263) / 3.141
   59 * 180
20 END

]RUN
-7.19834926
```

1-9 문자함수

마이크로 컴퓨터는 정보를 저장하는 고체전자 기억장치(solid-state electronic memory)와 계산, 의사결정, 정보이동 등을 수행하는 **중앙 연산 처리장치** (Central Processing Unit)*로 구성된다.

전기는 ON과 OFF의 두가지 기본적인 상태를 가졌기 때문에 2진수로 표현하는 것이 이상적일 것이다. 0혹은 1의 2진숫자 하나를 "비트(bit)"라고 부르며, 여덟개의 비트가 한 "바이트(byte)"를 형성한다. 정수들이나 한 문자를 저장하는 데는 기억장치가 한 바이트가 필요하다. 기억장치는 CPU의 명령들이나 데이타를 모두 2진수로 저장한다.

컴퓨터는 문자나 특수기호 외에 숫자들도 저장할 수 있을때 그 성능이 더욱 크게 증가되는 것이다. 이를 위해서 문자나 특수기호를 2진형태로 저장될 수 있는 숫자로 바꾸기 위한 코드가 마련됐다. 이 코드는 컴퓨터회사들에 의해 **미국표준정보코드(ASCII)**로 표준화 되었다. 이 코드에는 알파벳 소문자, 대문자, 여러기호및 컴퓨터에서 사용되는 각 제어기능들이 하나의 숫자로 표현되었다.

예로서 "JIM"은 컴퓨터에 의해 ASCII수인 74, 73, 77로 번역된다. 다시말해서 이 숫자들은 컴퓨터에 2진수형으로 저장된다.

J=74=01001010
I=73=01001001
M=77=01001101

ASC

*註
컴퓨터의 핵심부분으로 연산, 판단, 제어기능을 수행하며 컴퓨터내의 모든 데이터의 처리가 이곳에서 이루어진다.



ASC

컴퓨터가 JIM같은 알파벳이나 비수치적 정보를 저장하기 위해서는 그것들은 문자의 형태로 입력되어야 하며, 문자는 하나하나 ASCII숫자로 바뀌어 2진수로 기억장치에 저장된다. 표 1 - 2는 Applesoft에서 사용되는 ASCII문자 코드표다.

표1-2 ASCII문자 CODE

10진	16진	문자명	키보드	10진	16진	문자명	키보드	10진	16진	문자명	키보드
0	00	NULL	ctrl @	32	20	SPACE	SPACE	64	40	@	@
1	01	SHO	ctrl A	33	21	!	!	65	41	A	A
2	02	STX	ctrl B	34	22	"	"	66	42	B	B
3	03	ETX	ctrl C	35	23	#	#	67	43	C	C
4	04	ET	ctrl D	36	24	$	$	68	44	D	D
5	05	ENQ	ctrl E	37	25	%	%	69	45	E	E
6	06	ACK	ctrl F	38	26	&	&	70	46	F	F
7	07	BEL	ctrl G	39	27	'	'	71	47	G	G
8	08	BS	ctrl H or←	40	28	(	(	72	48	H	H
9	09	HT	ctrl I	41	29	)	)	73	49	I	I
10	0A	LF	ctrl J	42	2A	*	*	74	4A	J	J
11	0B	VT	ctrl K	43	2B	+	+	75	4B	K	K
12	0C	FF	ctrl L	44	2C	,	,	76	4C	L	L
13	0D	CR	ctrl M or RETURN	45	2D	–	–	77	4D	M	M
14	0E	SO	ctrl N	46	2E	.	.	78	4E	N	N
15	0F	SI	ctrl O	47	2F	/	/	79	4F	O	O
16	10	DLE	ctrl P	48	30	0	0	80	50	P	P
17	11	DC1	ctrl Q	49	31	1	1	81	51	Q	Q
18	12	DC2	ctrl R	50	32	2	2	82	52	R	R
19	13	DC3	ctrl S	51	33	3	3	83	53	S	S
20	14	DC4	ctrl T	52	34	4	4	84	54	T	T
21	15	NAK	ctrl U or→	53	35	5	5	85	55	U	U
22	16	SYN	ctrl V	54	36	6	6	86	56	V	V
23	17	ETB	ctrl W	55	37	7	7	87	57	W	W
24	18	CAN	ctrl X	56	38	8	8	88	58	X	X
25	19	EM	ctrl Y	57	39	9	9	89	59	Y	Y
26	1A	SUB	ctrl Z	58	3A	:	:	90	5A	Z	Z
27	1B	ESCAPE	ESC	59	3B	;	;	91	5B	[	[
28	1C	FS	n/a	60	3C	<	<	92	5C	\	\
29	1D	GS	ctrl shift-M	61	3D	=	=	93	5D	]	]
30	1E	RS	ctrl ∧	62	3E	>	>	94	5E	∧	∧
31	1F	US	n/a	63	3F	?	?	95	5F	–	–

※ n/a는 key board에 없는 것

그럼 BASIC에서의 ASCII코드 사용법에 대해 알아 보기로 하자.
문자를 숫자로 변환하는 것은 ASCII코드를 사용함으로써 이루어진다.

ASC(X$)은 문자열 X$의 선두 한문자의 ASCII코드가 10진수로 변환한 수를 나타내 준다.

```
]X$="A":PRINT ASC(X$)
65
```

위의 예에서 A는 ASCII코드로 65이다.

반대로 수를 문자로 바꾸는데는 CHR$(X)를 사용한다. 이것은 10진수 X에 해당하는 ASCII코드의 문자를 나타내준다. X값이 0~255의 범위가 아닐 때는 물론 CHR$

```
]PRINT CHR$(256)

?ILLEGAL QUANTITY ERROR
```

가 표시되고 에러가 된다.

CHR$(X)를 사용하여 ASCII코드표와 같은 문자, 마크, 특수기능을 갖게하는 것이 가능하다. 예를들면 CHR$(4)는 콘트롤(CTRL)D로 DOS명령을 뜻한다. DOS는 Disk Operating System의 약칭이다.

다음은 CHR$(X)를 사용하여 문자및 기호를 프린트하는 프로그램이다.

```
* PROGRAM  1.44
 10  FOR I = 32 TO 126
 20  PRINT I;"= "; CHR$ (I);" ";
 30  NEXT I
 40  END
```

이 프로그램을 좀더 보강하여 EPSON PRINTER을 이용하여 ASCII CODE를 나타내도록 프로그램을 작성하면 다음과 같다.

```
* PROGRAM  1.45
 1  REM
       ASCII CODE
 2  REM
 10  PRINT
 20  PRINT  TAB( 15);"ASCII  CODE"

 30  PRINT  TAB( 14);"------------
      -": PRINT  TAB( 22);"PRINTER
      :EPSON RP-80"
 40  PRINT "-----------------------
      ------------------"
 50 A$ = "DEC CHAR."
 60  PRINT  TAB( 1);A$; TAB( 17);A
      $; TAB( 33);A$
 70  PRINT "-----------------------
      ------------------"
 80  FOR I = 32 TO 127
 90  PRINT I;"   "; CHR$ (I),
 100  NEXT I
 110  PRINT "---------------------
      ------------------"

]RUN

                 ASCII  CODE
                -------------
                     PRINTER:EPSON RP-80
-----------------------------------------
DEC CHAR.        DEC CHAR.        DEC CHAR.
-----------------------------------------
32               33   !           34   "
35   #           36   $           37   %
38   &           39   '           40   (
41   )           42   *           43   +
```

44	,	45	-	46	.
47	/	48	0	49	1
50	2	51	3	52	4
53	5	54	6	55	7
56	8	57	9	58	:
59	;	60	<	61	=
62	>	63	?	64	@
65	A	66	B	67	C
68	D	69	E	70	F
71	G	72	H	73	I
74	J	75	K	76	L
77	M	78	N	79	O
80	P	81	Q	82	R
83	S	84	T	85	U

86	V	87	W	88	X
89	Y	90	Z	91	[
92	¥	93	]	94	^
95	_	96	'	97	a
98	b	99	c	100	d
101	e	102	f	103	g
104	h	105	i	106	j
107	k	108	l	109	m
110	n	111	o	112	p
113	q	114	r	115	s
116	t	117	u	118	v
119	w	120	x	121	y
122	z	123	{	124	¦
125	}	126	~	127	

STRING

컴퓨터는 문자나 기호도 다룰 수 있다. 그것도 1문자 뿐이 아니라 일련의 문자도 취급할 수 있다. 이것은 우리들이 문자를 많이 사용하고 있는 것과 같다.

문자의 배열을 「literal」이라고 하고 literal을 쿼트로 묶은 것을 스트링(string)이라고 한다. 예를들면,

```
"APPLE"
"KWACK"
"THIS IS A TEST"
```

등은 모두 스트링이다.

수변수와 마찬가지로 문자열을 포함한 스트링변수에도 어느 특정한 값을 대입할 수 있으며 반드시 스트링 변수는 변수명뒤에 달러마크$를 붙여 일반 변수와 구별하도록 되어 있다.

```
A$
NAME$
SENTENCES$
```

변수 A와 변수 A$은 서로 다르므로 이 Z변수는 동일 프로그램에서 사용할 수 있다.

지금 「HONG KIL DONG」이라는 문자열을 NAME$라는 스트링 변수에 넣고 싶을 때는 다음과 같이 한다.

```
]10 NAME$="HONG KIL DONG"

]20 PRINT NAME$
```

라는 명령을 실행시키면 NAME$의 내용이 표시될 것이다. 물론 문자는 쿼트로 묶어야 한다. 이와 같이 자주 사용하는 문자 스트링은 짧은 이름을 붙여서 변수로서 지정해 두면 편리하다.

LEN

BASIC에는 스트링을 더 편리하게 다루는 인스트럭션이 몇개 더 있다. 예를들

면 스트링의 길이 즉 몇문자를 포함하고 있나를 알고 싶을 때는

```
]PRINT LEN("HONG KIL DONG")
```

으로 하거나 또 같은 스테이트먼트로서 다음과 같이도 할 수 있다.

```
]NAME$="HONG KIL DONG":PRINT LEN(NAME$)
```

이렇게 하면 컴퓨터는 스트링의 길이(length)를 인자한다. 이 경우 13이 된다. 스페이스도 문자로서 세어지는 것에 주의하자.

이와 같이 스트링 변수의 길이를 알려면 LEN함수를 사용한다.

```
LEN (X$)
```

는 X$의 문자의 수를 나타낸다.

스트링에서 다루어지는 문자수는 0～255까지이다. 만일 스트링중에 255 이상의 문자수를 사용했을 경우

```
?SYNTAX ERROR
```

또는

```
?STRING TOO LONG ERROR
```

의 메세지가 나온다. 문자 제로의 스트링을 널(null) 스트링이라고 한다.

다음을 보자.

```
* PROGRAM  1.46
]10 A$=" GOOD MORNING"
]20 PRINT A$

]RUN
 GOOD MORNING
```

즉 "GOOD MORNING"이라는 값이 A$에 대입된 것이 된다. 단 PRINT$와 같이 BASIC 명령이나 스테이트먼트로 쓰이고 있는 예약어(reserved word) 에 $를 붙이거나 AFOR$와 같이 예약어(여기서는 FOR) 를 포함해서는 안된다.



LEN

LEFT$

스트링변수 중에서 어느 특정한 부분만을 뽑아내어 그 부분에 조작을 행할 때가 종종 있다. 이 경우에는 다음 3종류의 편리한 명령이 있다. 즉 LEFT$, RIGHT$ 및 MID$이다.

예를 들면 「GOOD MORNING」이라는 값이 A$에 들어있다고 하고 이 스트링 최초의 4개의 문자만을 뽑아내려고 할 때에는 다음과 같이 한다.

```
* PROGRAM  1.47
]10 A$="GOOD MORNING"

]20 PRINT LEFT$(A$,4)

]RUN
GOOD
```

LEFT$이라는 함수는 주어진 스트링 변수 중에서 주어진 수만큼 좌측부터 문자를 취해서 서브스트링을 만들어내는 작용을 한다.

RIGHT$

LEFT$에 대응해서 RIGHT$이라는 함수도 있다. 이것은 LEFT$의 기능에서 문자를 우측부터 취하는 것으로 바꾸었을 뿐이다.

MID$

그러면 가운데 문자를 꺼집어 내려면 어떻게 하면 좋을까?

MID$ (A$, N, M)을 사용해 보자. 이것은 A$의 문자열중 좌측에서 세어서 N번째부터 M개의 문자열을 추출하도록 한다.

다음 예를 보자.

```
* PROGRAM  1.48
]10 READ A$,B$

]20 PRINT MID$(A$,8,7)

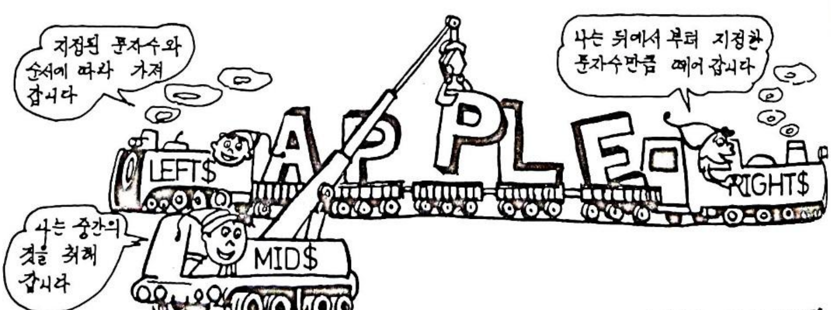
]30 A$=LEFT$(A$,7)+B$

]40 PRINT A$

]50 DATA "I AM A TEACHER","PIONEER"

]60 END

]RUN
TEACHER
I AM A PIONEER
```



LEFT$, MID$, RIGHT$

행번호 30에서는 "I AM A"에다 +기호에 의해 B$문자인 "PIONEER"가 연결되어 A$에 저장된다. 이 프로그램중 A$나 B$의 문자수보다 다음의 뽑아내려는 문자 치수가 더 큰 경우에는 A$, B$그 자체가 되고 문자수치가 0일때는 널스트링 (null string)이 된다.

```
]10 X$="GOOD MORNING":FOR N=1 TO LEN(X$):PRINT LEFT$(X$,N):NEXT N
```

를 RUN시켜 보면

```
]RUN
G
GO
GOO
GOOD
GOOD
GOOD M
GOOD MO
GOOD MOR
GOOD MORN
GOOD MORNI
GOOD MORNIN
GOOD MORNING
```

X$의 길이는 12이므로, 이 루프는 12번만 돌게 되며 한번 돌 때마다 N의 수만큼 좌측에서 인자를 행하고 있다.

여러분은 다음과 같은 치환문에서 하나의 스트링을 다른 것으로 바꾸어 놓을 수 있다.

```
X$ = A$
```

이 스테이트먼트는 A$의 내용을 그대로 X$에 복사한다. 단 이 스테이트먼트의 좌변에 스트링 부분을 나타내는 방법을 취할 수 없다. 예를들면

```
MID$ (X$,3,3) = "XYZ"
```

는 불가능 하나

```
X$ = MID$(A$,3,3)
```

는 된다. 치환문의 좌변은 요컨대 변수만이 사용 가능하게 된다.

자, 당신의 이름을 반대로 프린트하고 싶지 않는가?

```
* PROGRAM  1.49
 10  REM  ** PROGRAM TO SPELL YOUR
       NAME BACKWARDS **
 20  INPUT "TYPE YOUR NAME & I WIL
       L SHOW IT TO YOU SPELLED BAC
       KWARDS   ";N$
 30  REM  REVERSE ORDER OF LETTERS
 40  FOR T =  LEN (N$) TO 1 STEP  - 1
 50 R$ = R$ + ( MID$ (N$,T,1))
 60  NEXT T
 70  PRINT : PRINT "YOUR NAME SPEL
       LED BACKWARDS IS  ";R$
 80  PRINT : PRINT
 90  GOTO 20
```

CLEAR

몇개의 이름을 넣으면서 이 프로그램을 RUN해 보아라. 계속해 보면 무엇인지 잘못된 곳이 있다. 행번호 50이 문제의 곳이다.

```
]RUN
TYPE YOUR NAME & I WILL SHOW IT TO YOU SPELLED BACKWARDS   KWACK

YOUR NAME SPELLED BACKWARDS IS  KCAWK


TYPE YOUR NAME & I WILL SHOW IT TO YOU SPELLED BACKWARDS   PARK

YOUR NAME SPELLED BACKWARDS IS  KCAWKKRAP
```

RUN상태에서 본다면, KWACK을 입력했을 때 N$는 KWACK으로, R$는 KCAWK로 될 것이다. 다음에 PARK을 넣으면 N$는 PARK로 된다. 이어서 행번호 50은 R$를, 오래된 R$에 N$의 반대철자를 더한 KCAWKKRAP로 해 버린다. 그래서 스트링 변수를 0에 리셋하는 커맨드가 필요하다. 그렇게 하면 GOTO뒤에 R$에 새로운 문자가 들어간다.

다행히도 BASIC에는 이와 같은 명령이 있다. 이것은 CLEAR이다.

CLEAR는 모든 변수를 리셋해 버린다. 이것을 가해 보자.

```
]85 CLEAR
```

자, 프로그램을 다시 RUN시켜 보자.

CLEAR명령은 직접명령으로서도 사용가능하다.

```
]N=254

]PRINT N
254
```

로 하고, 다음에

```
]CLEAR
```

로 하고, 다시 PRINT N으로 하면

```
]PRINT N
0
```

N의 값은 0으로 되었다.

STR$

수와 그것을 표시한 스트링을 서로 변환하고 싶을 때가 있는데 이때는 스트링화 함수 STR$ 및 수치화함수 VAL을 사용한다.

STR$(X)는 수치X가 숫자로서의 의미를 잃고 문자열로 바꾸어 주며 VAL(X$)의 반대이다. 즉 수로서 567과 문자로서의 567은 의미가 다르다.

그러므로 수로서의 567에 STR$를 취하면 문자로서의 567로 된다.

```
]STRING$ = "567"
]PRINT VAL(STRING$)
567
```

STR은 string(文字列)의 약자이다.

```
]STRING$ = STR$(3.14159265)

]PRINT STRING$,LEFT$(STRING$,5)
3.14159265       3.141
```

STR$함수는 수치를 포맷에 적용시킬 때에도 사용할 수 있다.
다음 프로그램을 검토해 보자.

```
* PROGRAM  1.50

10  INPUT X,Y
20 A$ =  STR$ (X)
30 B$ =  STR$ (Y)
40 A1$ =  RIGHT$ (A$,1)
50 B1$ =  LEFT$ (B$,1)
60 S$ = A1$ + B1$
70  PRINT "ABC= ";X,"DEF= ";Y
80  PRINT  VAL (S$)
90  END
```

이 프로그램은 세자리의 양의 정수 ABC와 DEF를 입력하여 CD라는 두 자리의 수를 만드는 것이다.

이 프로그램을 RUN시키면 행10의 INPUT문에 의해 X, Y의 입력을 기다린다. X,Y의 각각의 값을 357, 829라 하면

```
]RUN
?357,829
ABC= 357         DEF= 829
78
```

로 실행된다. 여기서 행20은 입력된 수치를 ABC문자로서 읽고 행30에서 입력된 수치를 DEF문자로서 읽는다.

행 40과 50에서 C와 D를 골라내어 행60에서 합치고 행70은 입력된 ABC와 DEF를 프린트하고 행 60에서 만들어 낸 CD문자를 숫자로 바꾸어 78이라고 프린트하였다.

한편 STR$은 인쇄할 때에 몇자리가 필요한지를 조사할 때에도 사용된다. 예를들면,

```
]PRINT LEN (STR$(33333.157))
9
```

와 같이 33333.157은 9자리가 필요한 것으로 나타났다.

앞에서 잠깐 비친 스트링의 연결에 대해 알아보자.

당신의 컴퓨터로 다음을 시도해라.

```
]C$="GOOD MORNING"

]D$=C$+" "+"KIM"

]PRINT D$
```

당신의 컴퓨터는

```
GOOD MORNING KIM
```

라고 납할 것이다. 이러한 플러스(+)기호를 사용하여 연결하는 방법은 스트링을 나누거나 함께 하거나를 약간의 변경으로 가능케 하는 점에서 특히 유용하다. 예를 들면 위의 D$와 같으나 이것에 대시(-)를 붙인 스트링을 새로이 만들려면

```
]E$=RIGHT$(D$,4)+"-"+LEFT$(D$,4)+"-"+MID$(D$,6,7):PRINT E$
```

로 하면

```
KIM-GOOD-MORNING
```

로 되어 표시된다.

스트링은 수를 포함한 거의 모든 문자를 다룰수 있다. 그러나 PRINT문과 마찬가지로 스트링 중에 쿼트로 둘러싸인 문자는 비록 그들이 수더라도 산술적으로는 취급할 수 없다.

```
]C$ = "123"

]PRINT C$ + 4
```

은 다음과 같은 메세지를 낸다.

```
?TYPE MISMATCH ERROR
```

그리고 최후의 스테이트먼트를 실행할 수 없다. 여기서 이 기능을 사용하자.

VAL 이것은 value기능으로 스트링의 내용을 대응하는 수치로 바꾼다.

```
]PRINT VAL(C$) + 4
```

에서는 명백히 되고 연산결과가 127이 표시되어 온다. 여기서 VAL을 사용할때는 스트링 변수를 괄호로 둘러싸는 것을 잊어서는 안된다.

감산의 경우도 같은 취급으로

```
]Q = VAL(C$) - 21

]PRINT Q
102
```

가 된다. 또 2개의 다른 스트링의 VAL에 의한 덧셈도 가능하다.

```
]K$ = "12"

]P = VAL(C$) + VAL(K$)

]PRINT P
135
```

VAL(X$)는 문자열 X$의 첫문자가 수치일때만 그 문자열이 표시하는 수치를 나타낸다. 이때 수치 앞에 +, -의 부호가 있어도 무관하다. 이 외에는 모두 0로 된다.

어느 경우에는 수를 문자로 하고 싶을 때도 있다. 이것은 STR$라는 기능으로 VAL과는 반대의 관계가 된다. P라는 수변수를 스트링변수로 할 때는

```
]P$ = STR$(P)

]PRINT P$
135
```

로 하면 스트링으로서의 P를 표시한다. 자, 여기서 이 장에서 배운 것을 정리해 보자.

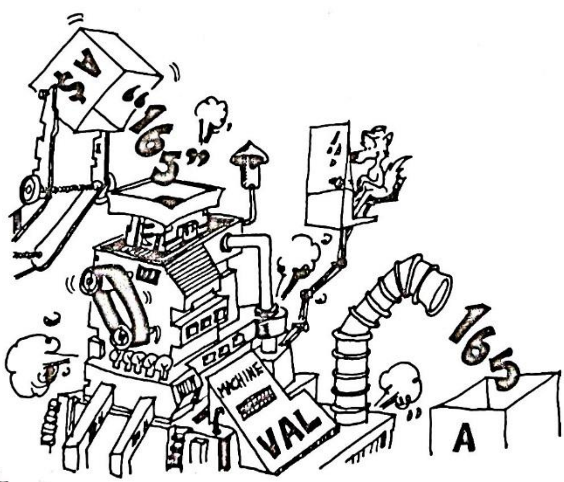
```
* PROGRAM 1.51

10  INPUT "TYPE A NUMBER FROM 1 T
    HROUGH 999999999? ";N$
20 N =  VAL (N$)
30  IF N < 1 OR N > 999999999 THEN
     GOTO 10
40 N$ =  STR$ (N)
50  FOR T =  LEN (N$) TO 1 STEP
    1
60 P$ = P$ +  MID$ (N$,T,1)
70  NEXT T
80  PRINT : PRINT "ORIGINAL",N$
90  PRINT : PRINT "REVERSED",P$
100 P =  VAL (P$)
110  PRINT : PRINT "ORIGINAL + RE
     VERSED = ";N + P
120  CLEAR
130  PRINT : PRINT : GOTO 10
```

이 프로그램은 행 10에서 1∼999999999 범위의 수를 입력시키면 입력된 수와 그 수의 발전된 수를 프린트하고 그 합을 계산하는 것이다. 자, RUN시켜 실행시켜 보자.

```
]RUN
TYPE A NUMBER FROM 1 THROUGH 999999999? 1234

ORIGINAL        1234
```



VAL

```
REVERSED          4321

ORIGINAL + REVERSED = 5555


TYPE A NUMBER FROM 1 THROUGH 999999999? 1001

ORIGINAL          1001

REVERSED          1001

ORIGINAL + REVERSED = 2002
```

문제82. "I AM A BOY."
와 같이 인쇄하는 프로그램을 작성하라. 꼭 따옴표는 표시되어야 한다.

```
10 X$ = CHR$ (34)
20 PRINT X$;"I AM A BOY.";X$
30 END

]RUN
"I AM A BOY."
```

문제83. 다음의 결과와 같은 프로그램을 작성하라.

```
]RUN
Apple
```

```
10 PRINT CHR$ (65);
20 PRINT CHR$ (112);
30 PRINT CHR$ (112);
40 PRINT CHR$ (108);
50 PRINT CHR$ (101);
60 END
```

문제84. DATA문을 이용하여 ASCII수로서 ASCII DID THIS!를 출력하는 프로그램을 작성하라.

```
10 DATA 65,83,67,73,73,32,68,73
     ,68,32
20 DATA 84,72,73,83,33
30 FOR I = 1 TO 15
40 READ X
50 PRINT CHR$ (X);
60 NEXT I

]RUN
ASCII DID THIS!
]
```

문제85. 아래의 전화번호 목록을 읽고 44국만 추출하여 인쇄하는 프로그램을 작성하라.

12－4567, 34－5436, 78－1234, 44－1234, 56－8907, 44－0987, 56－5640, 41－3400, 40－2087, 66－9993

```
10 FOR I = 1 TO 10
20 READ X$
30 IF VAL (X$) = 44 THEN PRINT
     X$
40 NEXT I
50 END
60 DATA "12-4567","34-5436","78-
     1234","44-1234","56-8907"
70 DATA "44-0987","56-5640","41-
     3400","40-2087","66-9993"

]RUN
44-1234            44-0987
```

문제86. 400m 릴레이에서의 네 사람의 기록은 각각 11.3, 11.1, 10.9, 12.0초 이었다. 합계 시간을 인쇄하는 프로그램을 작성하라.

```
10 PRINT STR$ (11.3 + 11.1 + 10
   .9 + 12.0);" ";"SECOND"
20 END

]RUN
45.3 SECOND
```

문제87. 아래의 수를 입력하여 "342617"으로 인쇄하는 프로그램을 작성하라.

DATA 34, 26, 17

```
10 S$ = ""
20 FOR I = 1 TO 3
30 READ X
40 A$ = RIGHT$ ( STR$ (X),2)
50 S$ = S$ + A$
60 NEXT I
70 PRINT S$
80 END
90 DATA 34,26,17

]RUN
342617
```

문제88. VAL를 사용하여 "1988-SEOUL OLYMPIC"을 프린트하는 프로그램을 작성하라.

```
10 X$ = "1988-SEOUL OLYMPIC"
20 IF VAL (X$) < > 0 THEN PRINT
   X$

]RUN
1988-SEOUL OLYMPIC
```

문제89. 문자열을 2개 입력하여 일치하면 그 문자열을, 일치하지 않으면 2개 모두 프린트하는 프로그램을 작성하라.

```
10 INPUT X$
20 INPUT Y$
30 IF X$ = Y$ THEN PRINT X$: GOTO
   50
40 PRINT X$;"  ";Y$
50 END

]RUN
?KIM
?PARK
KIM  PARK
```

문제90. 60이상 95이하까지의 난수 20개를 취한 후, 이 수에 대응되는 ASCII문자를 출력하는 프로그램을 작성하라.

```
10 FOR X = 1 TO 20
20 C = RND (1) * 26 + 65
30 PRINT CHR$ (C);
40 NEXT X

]RUN
HVDVOXLADZDZSZUHSTCW
]
```

문제91. ASCII코드표를 이용하여 다음 프로그램의 출력을 구해 보고 실제로 프로그램을 수행한 결과와 비교해 보라.

```
10 FOR A = 1 TO 3
20 READ A$
30 PRINT ASC (A$),
40 NEXT A
50 DATA "#","$","&"
```

```
]RUN
35            36            38
```

문제92. 10진수(Decimal)를 16진수(Hex)로 변환하는 프로그램을 작성하라.

```
1  REM
      DECIMAL TO HEX
2  REM
10 H$ = "0123456789ABCDEF"
20 BASE = 16:I = 1
30  INPUT "NUMBER (DECIMAL) ";N:N
      =  INT (N)
40 Q =  INT (N / BASE)
50 A(I) = N - BASE * Q:I = I + 1
60  IF Q > BASE THEN N = Q: GOTO
     40
70 A(I) = Q
75  PRINT
80  PRINT  TAB( 9);"$";
90  FOR J = I TO 1 STEP  - 1
100  PRINT  MID$ (H$,A(J) + 1,1);
110  NEXT J
115  PRINT

]RUN
NUMBER (DECIMAL) 8192

        $2000

]RUN
NUMBER (DECIMAL) 1262

        $4EE
```

문제93. 16진수를 10진수로 바꾸는 프로그램을 작성하라.

```
1  REM
      HEX TO DECIMAL
2  REM
10  INPUT "NUMBER (HEX)?  ";A$
20 L =  LEN (A$):N = 0
30  FOR I = 0 TO L - 1:T =  ASC (
     MID$ (A$,L - I,1))
40  IF T = 36 THEN  GOTO 80
50 T = T - 48: IF T > 9 THEN T =
    T - 7
60 N = N + T * 16 ^ I
70  NEXT I
75  PRINT
80  PRINT  TAB( 15);"$";A$;" = ";
    N

]RUN
NUMBER (HEX)?  7FF

              $7FF = 2047

]RUN
NUMBER (HEX)?  ABCD

              $ABCD = 43981
```

1-10 배 열

DIM

배열(array)이란 수표(數表)를말한다. 배열을 사용하면 수표중의 어느 요소라도 꺼낼 수 있으며, 이 표의 이름은 배열명이라고 해서 일반 변수명이 사용된다. 예를 들어 A라면 배열명 A는 단순한 변수 A와는 명확하게 구별된다.

수치 계산에만 국한하지 않으나 예로서 행렬식의 각 요소와 같이 여러가지 값으로 서로 다른 데이타에서는 어떤 동일 종류의 데이타로 취급하는 경우가 있다. BASIC에서는 프로그램중에 어느 변수가 있으면 그 변수는 하나의 기억장소를 이용하는 것이 된다. 즉 배열을 만들어 내기 위해 우선 컴퓨터에 대해서 최대 몇

개의 요소가 되는지를 알리지 않으면 안된다. 이를 위해 DIM문이 사용된다.

DIM문(Dimension)은 1차원, 2차원 및 3차원뿐만 아니라 다차원을 최대 255×255의 범위내에서 배열할 수 있으며 여러개의 집단데이타를 하나의 변수명으로 등록하고 지정된 만큼의 메모리 영역을 확보하는 문이다.

배열 요소의 초기치는 0이다.

1차원의 수치 배열은

1차원 배열

DIM A(I)

의 형식으로 쓰인다. 이 형식에서 A와 I는

A: 수치배열명(문자 A로 꼭 국한하지 않는다)

I: 수치의 첨자로 I+1개의 기억장소를 확보

를 표시한다. 이를 표현하면 그림1-4와 같다.

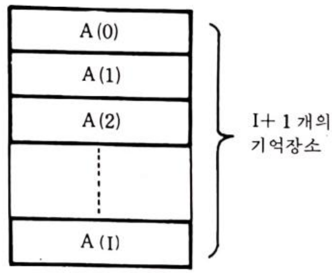


그림1-4 1차원의 배열

배열중의 각 요소는 0번에서 시작되므로 배열명A가 5개의 데이타 요소일때는

```
DIM(4)
```

가 된다. 이것으로 5의 새 변수가 예약된 것이 된다. 즉 메모리 영역에 변수

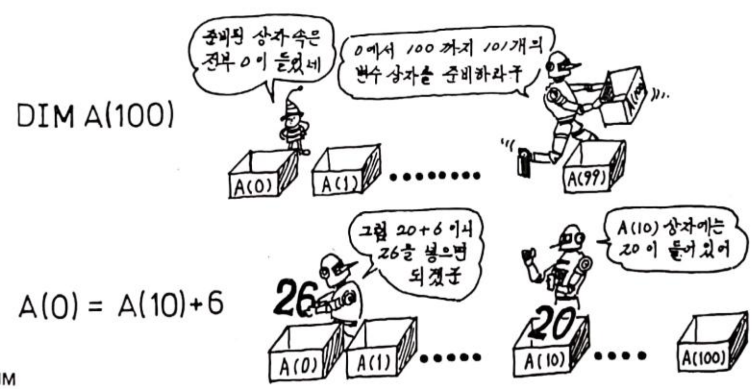
```
A(0)
A(1)
A(2)
A(3)
A(4)
```

를 위한 5개의 데이타 영역을 취하게 된다.

괄호안의 수는 첨자(添字, subscript)라고 하고 A(4)라는 첨자붙은 수치 변수를 "Asub 4"라고 읽는다.

첨자는 수식이나 변수인 경우도 있다.

```
A(2) = 4 + 3
```



DIM

이것은 A(2)라는 데이타영역, 쉽게 말하면 변수상자속에 수식 4＋3의 계산 결과를 넣어라는 뜻이며,

```
A(0) = A(4) + 8
```

은 A(4)에 들어 있는 수치에 8를 더한 값을 A(0)에 넣어라는 것이다.

DIM문이 실행되면 수치 배열은 모두 0로, 문자배열은 모두 null(길이가0)로 된다. 다시말하면 문자배열의 변수상자는 텅비어 있다는 것이다. 이 예를 한번 보자.

```
B$(11)="A"+"5"
```

이것은 스트링 변수의 경우로서 B$(11)중에 스트링은 "A"+"5", 즉 A5 인 것을 나타낸다.

```
A$(0)="Z"
A$(10)=A$(0)+"X"
```

이 예는 A$(0)속의 스트링은 Z이며, A$ (0)에다 X를 더하여 ZX를 A$(10)의변수 상자에 넣으라는 뜻이다.

DIM문은 메모리 영역을 확보하는 외에 2차원 배열과 행열의 요소의 수도 지정한다.

2차원 배열 이 경우, 2차원의 수치배열은

DIM A(I, J)

의 형식으로 쓰여진다. 이 형식에서 A와 I, J는

A : 수치배열명

I, J : I와J는 수치이고 I＋1행×1행×J＋1열의 기억장소를 확보

를 표시한다. 이를 도시하면 그림 1－5와 같다.

예로서

```
DIM A(2,2),B(3,3)
```

여기서 A에는 9개의 요소 A(0, 0), A(0, 1), ……, A(2, 2), B에는 16개의 요소 B(0, 0), B(0, 1), ……, B(3, 3)가 확보된다. 각 상자내의 초기치는 프로그램의 특정한 값이 지정되기 전까지는 0이다.

수치변수의 차원수는 최고 88차원까지 지정할 수 있다. 그러나 스트링변수는 1차원 일때는 사용되나, 대부분의 BASIC시스템에서는 2차원 스트링변수는 허용되지 않는다.

3차원 배열 3차원 배열의 수치 배열은

DIM A (H, W, L)

의 형식으로 표시된다. 첨자 H, W, L은 수치이고

(H＋1)×(W＋1)×(L＋1)개

의 기억장소가 확보된다. 이것을 H＝0, W＝2, L＝3인 경우에 대해서 그림1－6과 같이된다.

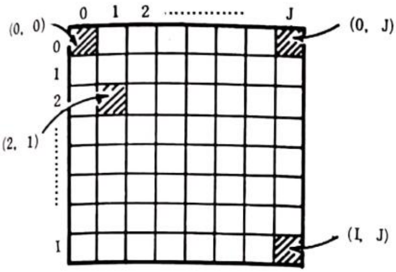


그림1-5 DIM A(I, J)

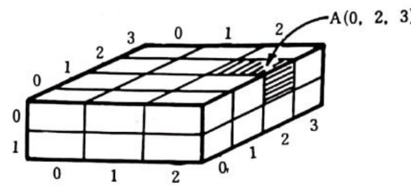


그림1-6 A(H, W, L) — A(0, 2, 3)의 예

실제로는 배열의 크기는 메모리수에 제약되는 일이 많으며, 예를들어 정수의 배열에서 1개의 변수는 2바이트(byte), 실수는 5바이트, 스트링 상수는 수변수 영역중에서 길이에 1바이트, 포인터로서 2바이트, 그리고 스트링 변수 영역에 길이분 만큼의 메모리를 사용하며 프로그램에 의해서 스트링 변수가 격납되었을때는 1문자에 대해 메모리를 1바이트 여분으로 사용한다.

배열이 프로그램중 DIM으로 선언하기 전에 사용되었을 때에는 자동적으로 각각의 차원에 대해서 11개의 데이타요소(첨자 0부터 10까지)를 확보하게 된다.

배열의 첨자가 선언한 값보다 클때나, 차원이 다른 배열로서 다루었을때, 예를들어 2차원 배열로서 정의 하였음에도 불구하고 3차원 배열로서 불러냈을때나 또는 음의 첨자로 사용했을 때는

```
?BAD SUBSCRIPT ERROR IN XX
```

라는 에러메세지를 발생시킨다.

이미 선언되어 있는 배열에 대해 같은 이름으로 다시 배열선언을 했을 때는

```
?REDIM'D ARRAY ERROR
```

가 표시된다.

또 스트링 배열내의 개개의 스트링 변수는 그 크기를 선언할 필요가 없다. 컴퓨터는 변수의 동적 재배열을 행하고 있으므로 스트링 변수는 늘었다 줄었다한다.

```
]WARD$(5)="ABCDE"
```

라는 문은 길이가 5이고 그 값이 ABCDE라는 스트링 변수를 만들어 내는데 그 후에

```
]WARD$(5)=""
```

로 하면 WARD $(5)에 할당된 영역은 없어진다. 스트링 변수는 최대 255 문자의 길이를 취할 수 있다.

배열의 요소는 RUN 또는 CLEAR에 의해서 모두 그 값은 0가 된다. 물론 앞

서 설명했듯이 수 변수는 0으로, 스트링 변수는 모두 널스트링이 된다.

```
* PROGRAM  1.52
 10  DIM A(10)
 20  FOR I = 1 TO 10
 30 A(I) = I
 40  PRINT A(I);
 50  NEXT I
 60  CLEAR
 70  END

 ]RUN
 12345678910
```

이 프로그램중 행번호60에서 CLEAR는 모든 변수배열, 문자열을 0으로 바꾸어 주며 포인터와 스택을 reset시킨다.

문제94. 5×8의 매트릭스의 I행, J열의 값을 1의 자리를 I, 소숫점 이하 첫째자리를 J로 하는 표를 아래와 같이 써내는 프로그램을 작성하라.

```
]RUN
1.1 1.2  1.3  1.4  1.5  1.6  1.7  1.8
2.1 2.2  2.3  2.4  2.5  2.6  2.7  2.8
3.1 3.2  3.3  3.4  3.5  3.6  3.7  3.8
4.1 4.2  4.3  4.4  4.5  4.6  4.7  4.8
5.1 5.2  5.3  5.4  5.5  5.6  5.7  5.8
```

```
10  DIM X(5,8)
20  FOR I = 1 TO 5
30  FOR J = 1 TO 8
40 X(I,J) = I + J / 10
50  PRINT X(I,J); TAB( 5 * J);
60  NEXT J
70  PRINT
80  NEXT I
90  END
```

문제95. X (N)를 사용하여 위의 메트릭스를 작성하라.

```
10  DIM X(40)
20  FOR I = 1 TO 5
30  FOR J = 1 TO 8
35 N = 8 * (I - 1) + J
40 X(N) = I + J / 10
50  PRINT X(N); TAB( 5 * J);
60  NEXT J
70  PRINT
80  NEXT I
90  END
```

```
]RUN
1.1 1.2  1.3  1.4  1.5  1.6  1.7  1.8
2.1 2.2  2.3  2.4  2.5  2.6  2.7  2.8
3.1 3.2  3.3  3.4  3.5  3.6  3.7  3.8
4.1 4.2  4.3  4.4  4.5  4.6  4.7  4.8
5.1 5.2  5.3  5.4  5.5  5.6  5.7  5.8
```

문제96. 다음 출력이 나오도록 프로그램을 작성하라.

```
]RUN
1.1 1.2  1.3  1.4  1.5  1.6
2.1 2.2  2.3  2.4  2.5  2.6
3.1 3.2  3.3  3.4  3.5  3.6
4.1 4.2  4.3  4.4  4.5  4.6
5.1 5.2  5.3  5.4  5.5  5.6
6.1 6.2  6.3  6.4  6.5  6.6
7.1 7.2  7.3  7.4  7.5  7.6
8.1 8.2  8.3  8.4  8.5  8.6
9.1 9.2  9.3  9.4  9.5  9.6
```

```
1 P = 9:Q = 6
5 R = P * Q
10  DIM X(R)
20  FOR I = 1 TO P
30  FOR J = 1 TO Q
35 N = Q * (I - 1) + J
40 X(N) = I + J / 10
50  PRINT X(N); TAB( 5 * J);
60  NEXT J
70  PRINT
80  NEXT I
90  END
```

문제97. 구구단 표(multiplication table)를 만드는 프로그램을 DIM문을 이용하여 작성하라.

```
10  DIM A(9,9)
20  HOME
32  PRINT "  ** MULTIPLICATION TA
     BLE ** "
40  PRINT
50  FOR J = 1 TO 9
60  PRINT  TAB( 4 * (J - 1) + 5);
     J;
70  NEXT J: PRINT
80  PRINT "----------------------
     ------------------"
90  FOR M = 1 TO 9
100  PRINT M;" !";
110  FOR N = 1 TO 9
120 A(M,N) = M * N
130  PRINT  TAB( 4 * (N - 1) + 5)
     ;A(M,N);
150  NEXT N
160  PRINT
170  NEXT M
180  END

]RUN
  ** MULTIPLICATION TABLE **

    1   2   3   4   5   6   7   8   9
----------------------------------------
1 ! 1   2   3   4   5   6   7   8   9
2 ! 2   4   6   8   10  12  14  16  18
3 ! 3   6   9   12  15  18  21  24  27
4 ! 4   8   12  16  20  24  28  32  36
5 ! 5   10  15  20  25  30  35  40  45
6 ! 6   12  18  24  30  36  42  48  54
7 ! 7   14  21  28  35  42  49  56  63
8 ! 8   16  24  32  40  48  56  64  72
9 ! 9   18  27  36  45  54  63  72  81
```

1-11 커서 (Cursor) 의 위치

VTAB

VTAB는 커서의 절대위치를 지정하기 위한 명령 혹은 스테이트먼트이다. 이 스테이트먼트는 커서를 상하로는 움직이지만 좌우로는 움직일 수 없다. 스크린상에서는 제일 위의 행이 제 1행, 제일 아래의 행이 제 24행이므로 커서의 위치의 범위는 저절로 결정되며, 1≦커서의 위치≦24가 된다. 이 범위를 잃으면,

```
]VTAB 25

?ILLEGAL QUANTITY ERROR
```

의 에러 메세지가 표시된다.

VTAB는 TEXT 윈도우인가, TEXT나 그래픽모드에 관계없이 커서를 위치시킨다.

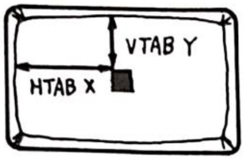
그래픽모드에서는 그래픽에리어에 커서를 가지고 간다.

윈도우가 설정되어 있을 때에 그 하한을 넘어서 VTAB가 사용되었을 때에는 계속해서 실행되는 PRINT문의 내용은 그 행에 PRINT를 넣는다.

HTAB

HTAB에서는 커서가 현재 위치하고있는 행이 255의 포지션을 가지고 있다고 가정한다. 커서가 위치하는 행은 1~40, 다음의 행이 41~80이라는 것이다. HTAB는 몇 번째 가로줄 인가의 지정된 위의 포지션으로 커서를 이동시킨다.

윈도우가 설정되어 있어도 HTAB는 그것을 뛰어 넘어서 커서의 이동을 하지만 윈도우의 범위를 잃고 커서를 움직여도 PRINT문의 최초의 문자가 PRINT 되면 커서는 원래의 위치로 되돌아간다.



HTAB, VTAB

커서의 위치는 화면의(X, Y)에 있게 된다.

커서가 현재 위치하는 행의 제일 좌측으로 커서를 되돌리려면

```
]HTAB 1
```

을 실행한다.

HTAB 0 는 256번째의 포지션으로 커서를 움직인다.

위치설정은 0부터 255까지의 수로 되어있고 이 범위를 놓치면

```
]HTAB256

?ILLEGAL QUANTITY ERROR
```

가 된다.

HTAB와 VTAB의 차이에 주목하라. VTAB에서는 24이상의 수를 설정하면 에러로 되었으나 HTAB에서는 40이상의 수를 설정해도 다음행으로 커서가 옮기므로 에러로는 되지 않는다.

TAB(

이와같은 명령으로는 또 TAB가 있다. TAB는 PRINT문에서 사용되어야 하며 위치설정은 괄호안에 들어있어야 한다. 예를 들면

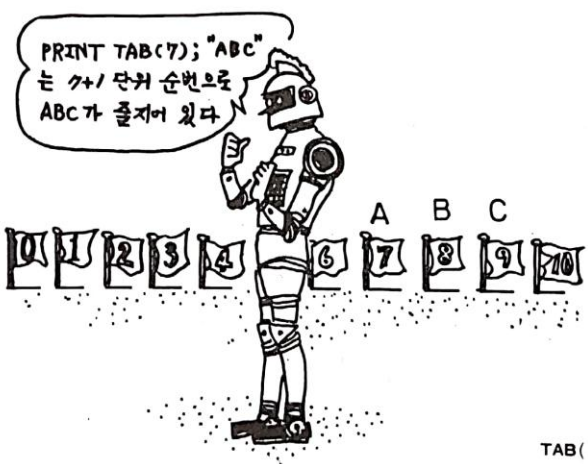
```
]PRINT TAB(5)"EXAMPLE"
```

는

```
␣␣␣␣ EXAMPLE
```

로 프린트된다. 여기서 ␣는 스페이스를 나타낸다.

TAB의 동작은 기본적으로 HTAB와 같으며, 현재 커서가 위치하는 행에서 인자 포지션이 1~40, 41~80, …와 같이 정해져 있고 위치설정에 따라서 커서를 움직이는데, 큰 차이는 현재의 커서의 위치보다 위치설정이 작은 경우, 커서는좌측으로 움직이지 않고 현재의 위치에서 움직이지 않는다는 것이다.



TAB(

이 경우 좌측으로 움직이려 할 때는 HTAB를 사용해야 한다. 그러나 TAB는 직접명령으로 움직이나 HTAB와 VTAB는 프로그램 중에서 밖에 사용할 수 없다.

```
]TAB(0)
```

는 커서를 포지션 256으로 옮긴다.

```
?ILLEGAL QUANTITY ERROR
```

가 되지 않기 위해 위치설정은 0 ≦위치설정≦255의 범위에 있어야 한다.

```
* PROGRAM  1.53
  10  FOR I = 1 TO 2
  20  READ X,Y
  30  PRINT  TAB( 10);X; TAB( 15);Y

  40  NEXT I
  50  DATA  456,79321,-56,-9321
  60  END
  70  END

  ]RUN
           456  79321
           -56  -9321
```

위의 예에서 수치 데이타가 양인 경우에는 부호가 생략되어 10번째와 15번째부터 인쇄하는 것으로 나타나나, 데이타가 음수인 경우에는 ㅡ부호가 10번째, 15번째 자리를 차지하게 된다.

POS

```
POS(0)
```

는 현재 커서가 위치하는 행의 좌단을 0으로서 커서의 위치를 준다. 다시 말하면 POS(0)는 현재의 커서의 위치를 알려주는 명령으로서 세로줄로 몇 번째 임을 숫자로 나타낸다. 다음 예를 보자.

```
* PROGRAM  1.54
  10  HTAB 20: VTAB 20
  20  PRINT "X"; POS (0)
  30  END

  ]RUN
                      X20
```

SPC(

HTAB 및 TAB에서는 커서의 위치를 1부터 세었으나 POS 및 SPC에서는 이것을 0부터 센다.

```
]PRINT TAB (23);POS (0)
                      22
```

는 22로 프린트되는데 대해서

```
]PRINT SPC (23);POS (0)
                     23
```

는 23으로 프린트된다.

```
SPC(X)
```

는 위치설정값인 X만큼 스페이스를 보내는 작용을 가지고 있다. TAB와 같이 SPC는 PRINT문 내부에서 사용되며 위치설정은 괄호속에 들어있어야 한다.

위치설정 X는

```
?ILLEGAL QUANTITY ERROR
```

를 막기위해 0~255의 범위이어야 한다. 그런데 255이상의 스페이스를 줄 수 없는 것이냐 하면 그렇지 않다. SPC를

```
]PRINT SPC(250) SPC(139) SPC(255)
```

와 같이 연속해서 사용함으로써 보다 많은 스페이스를 한 번에 줄 수 있다. HTAB가 현재 커서가 위치하는 행의 최초를 1로 하는 절대위치로 커서를 이동시키는데에 대해 SPC는 현재의 커서의 위치를 0으로 하는 상대위치로 이동하는 함수이므로 SPC(0)는 아무 변화도 스크린에 주지 않는다.

VTAB나 HTAB와 달리 SPC는 텍스트나 윈도우를 뛰어 넘을 수 없다. 항상 텍스트나 윈도우의 내부에서 스페이싱한다.

위치설정이 상수일 때는 SPC의 실행시에 정수로 변환된다.

INVERSE

만약 당신이 변함없는 검은 바탕에 흰 문자가 싫증이 난다면

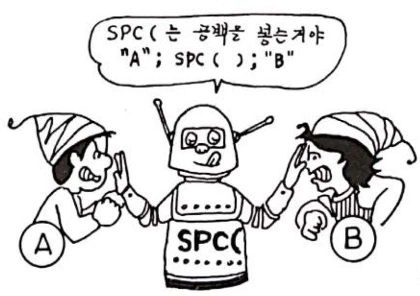
```
INVERSE
```

라고 타이프하고 프롬프트와 커서에 주목하라. 프롬프트는 흰바탕에 검은 것이 되었을 것이다. 지금 곧 간단한 프로그램을 타이프해 보아라.

```
]NEW

]100 PRINT "BLACK AND WHITE IN COLOR"
```

RUN시키면 모두 반전되어 나타날 것이다. 그러면



SPC

```
FLASH
```

FLASH

로 하고 RUN시켜 보자. 반짝반짝거리며 글씨가 점멸될 것이다.

INVERSE와 FLASH는 컴퓨터의 출력에만 유효하다. 타이프 했을 때에 표시되는 문자와는 관계가 없다. 이들 명령은 다이렉트나 프로그램에서라도 실행된다. INVERSE와 FLARH 커맨드를 여러가지 사용해 보면 검은 바탕에 흰 것이라도 그다지 지리하지 않다는 것을 알 것이다. 그렇다면

```
NORMAL
```

NORMAL

이라고 타이프하면 보통의 텍스트모드로 되돌아온다.

문제98. 다음과 같은 보고서 양식을 만들어 내기 위한 프로그램을 작성하라.

```
        SALES PEPORT

NO  COST       QTY   AMOUNT   TOTAL
```

```
10  PRINT
20  PRINT  TAB( 12);"SALES PEPORT
    "
30  PRINT
40  PRINT  TAB( 2);"NO"; TAB( 7);
    "COST";
50  PRINT  TAB( 17);"QTY"; TAB( 2
    3);"AMOUNT";
60  PRINT  TAB( 32);"TOTAL"
70  END
```

문제99. 5번째부터 ABC=, 10번째부터 −586.32, 20번째부터 DEF=, 25번째부터 93.153을 인쇄하는 프로그램을 작성하라.

```
10 A =  - 586.32
20 B = 93.153
30 PRINT  TAB( 5);"ABC="; TAB( 1
   0);A; TAB( 20);"DEF="; TAB(
   25);B
40 END

]RUN
    ABC= -586.32    DEF= 93.153
```

문제100. 오른쪽 그림과 같은 하트 모양의 출력을 얻기위한 프로그램을 TAB문을 이용해 작성하라.

```
]RUN
*****   *****
*    * *    *
*     *     *
*           *
 *         *
  *       *
    *   *
      *
```

```
5  PRINT
10  PRINT  TAB( 1);"*****"; TAB(
     9);"*****"
20  PRINT "*"; TAB( 6);"* *"; TAB(
     13);"*"
30  PRINT "*"; TAB( 7);"*"; TAB(
     13);"*"
40  PRINT "*"; TAB( 13);"*"
50  PRINT " *"; TAB( 12);"*"
60  PRINT  TAB( 3);"*"; TAB( 11);
     "*"
70  PRINT  TAB( 5);"*"; TAB( 9);"
     *"
80  PRINT  TAB( 7);"*"
90  END
```

문제101. 다음과 같은 결과가 되도록 프로그램을 작성하라.

```
]RUN
A     A
 B   B
  C C
   D
  E E
 F   F
G     G
```

```
10  FOR X = 1 TO 7
20  IF 8 - X = 4 THEN 100
30  IF 8 - X <  = X THEN 70
40  PRINT  TAB( X); CHR$ (64 + X)
    ;
```

```
50  PRINT  TAB( B - X); CHR$ (64 +
     X)
60  GOTO 110
70  PRINT  TAB( B - X); CHR$ (64 +
     X);
80  PRINT  TAB( X); CHR$ (64 + X)

90  GOTO 110
100  PRINT  TAB( X); CHR$ (64 + X
     )
110  NEXT X
120  END
```

문제 102. 다음의 결과와 같은 모양을 그리는 프로그램을 작성하라.

```
]RUN
##############################
##############################
##############################
##############################
##############################
##############################
##############################
```

```
10  FOR X = 1 TO 7
20  FOR Y = 1 TO 30
30  PRINT "#";
40  NEXT Y
50  PRINT
60  NEXT X
```

문제 103. 다음과 같은 모양을 그리는 프로그램을 작성하라.

```
]RUN
<><><><><><><><><><><><><><><>
><><><><><><><><><><><><><><><
><><><><><><><><><><><><><><><
><><><><><><><><><><><><><><><
```

```
10 A$ = "<"
20 B$ = ">"
30 C$ = A$ + B$
40 D$ = B$ + A$
50  FOR I = 1 TO 15
60  PRINT C$;
70  NEXT I
80  PRINT
90  FOR K = 1 TO 3
100  FOR L = 1 TO 15
110  PRINT D$;
120  NEXT L
130  PRINT
140  NEXT K
150  END
```

문제 104. $Y = X^2 - 2X + 1$ 의 그래프를 그리는 프로그램을 작성하라.

```
5  HOME
10  FOR I = 0 TO 35 STEP 5
20  HTAB (I + 3): VTAB 1
30  PRINT I;
40  NEXT I
50  FOR J = 3 TO 39
60  HTAB J: VTAB 2
70  PRINT  CHR$ (43)
80  NEXT J
90  FOR K = 3 TO 16
100  HTAB 3: VTAB K
110  PRINT  CHR$ (43)
120  NEXT K
130  FOR L =  - 6 TO 7
140  HTAB 1: VTAB (L + 9)
150  PRINT L;
160  NEXT L
170  HTAB 15: VTAB 10
180  PRINT "Y=X^2-2*X+1"
190  FOR X =  - 5 TO 7
200 Y = X ^ 2 - 2 * X + 1
210 A = 9 + X
220 B = Y + 3
230  HTAB B: VTAB A
240  PRINT "*"
250  NEXT X
260  END

]
```

```
 0  5   10  15  20  25  30  35
++++++++++++++++++++++++++++++++++++
-6+
-5+                                 *
-4+                       *
-3+              *
-2+        *
-1+   *
0 ++
1 +         Y=X^2-2*X+1
2 ++
3 +   *
4 +        *
5 +              *
6 +                       *
7 +                                 *
```

> **문제 105.** 비디오 스크린의 화면을 반으로 나누어 윗부분에 20개의 *와 같은 별들을 디스플레이하는 프로그램을 작성하라. 단 RND를 사용한 난수 표시일 것.

```
10  HOME
20  FOR N = 1 TO 20
30 X =  INT (40 *  RND (1) + 1)
40 Y =  INT (12 *  RND (1) + 1)
50  HTAB X
60  VTAB Y
70  PRINT "*"
80  NEXT N
```

```
 *
       *
                   *
                             *    *
                 **
    *               *
     *             *
                         *
  *                     *
*                *          *   *
                      *
                       *
```

> **문제 106.** SIN함수를 사용하여 다음 그림을 그리는 프로그램을 작성하라.

```
]RUN
                 KWACK!
                   KWACK!
                      KWACK!
                        KWACK!
                          KWACK!
                            KWACK!
                              KWACK!
                               KWACK!
                               KWACK!
                               KWACK!
                               KWACK!
                               KWACK!
                             KWACK!
                            KWACK!
                          KWACK!
                        KWACK!
                     KWACK!
                   KWACK!
                KWACK!
              KWACK!
           KWACK!
         KWACK!
       KWACK!
     KWACK!
   KWACK!
  KWACK!
  KWACK!
  KWACK!
  KWACK!
  KWACK!
    KWACK!
     KWACK!
       KWACK!
         KWACK!
            KWACK!
              KWACK!
```

```
10 RA = 3.14159 / 180
20  FOR I = 1 TO 360 STEP 10
30 T = I * RA
40 X =  SIN (T) * 15 + 18
50  HTAB (X): PRINT "KWACK!"
60  NEXT I
```

문제 107. Y=2X+1 의 그래프를 그리는 프로그램을 작성하라.

#50~#80 횡축을 +기호로서 그린다.

#90~#120 종축을 +기호로서 그린다.

#130~#160 횡축의 0에서 8까지의 눈금

#170~#180 Y=2X+1의 표시

#190~#260 Y=2X+1의 궤적을 X의 값을 0에서부터 7까지 0.5씩 변화시키면서 Y값을 구해서 *로써 프린트한다.

```
5  HOME
10  FOR I = 15 TO 0 STEP  - 1
20  HTAB 1: VTAB 17 - I
30  PRINT I
40  NEXT I
50  FOR J = 2 TO 16
60  HTAB 3: VTAB J
70  PRINT  CHR$ (43)
80  NEXT J
90  FOR K = 3 TO 19
100  HTAB K: VTAB 17
110  PRINT  CHR$ (43)
120  NEXT K
130  FOR L = 0 TO 8
140  HTAB (2 * L + 3): VTAB 18
150  PRINT L;
160  NEXT L
170  HTAB 6: VTAB 4
180  PRINT "Y=2X+1"
190  FOR X = 0 TO 7 STEP .5
200 Y = 2 * X + 1
210 A = 17 - Y
220 B = 2 * X + 3
230  HTAB B: VTAB A
240  PRINT "*"
250  NEXT X
260  HTAB 1: VTAB 19
270  END
```

```
15+                *
14+               *
13+   Y=2X+1     *
12+             *
11+            *
10+           *
9 +          *
8 +         *
7 +        *
6 +       *
5 +      *
4 +     *
3 +    *
2 +*
1 *
0 +++++++++++++++++
  0 1 2 3 4 5 6 7 8
```

문제 108. 컴퓨터로 하여금 다음의 출력이 나오도록 프로그램을 작성하라.

```
]RUN
1234567890123456789012345678901234567890

*     #       S     1      ( )       @
```

```
10  FOR I = 1 TO 4
20  PRINT "1234567890";
30  NEXT I
40  PRINT
50  PRINT "*"; TAB( 7);"#"; TAB(
   15);"S"; TAB( 21);"1"; TAB(
   28);"("; TAB( 30);")"; TAB(
   39);"@"

]RUN
1234567890123456789012345678901234567890

*     #       S     1      ( )        @
```

문제 109. 난수에 의한 곱셈으로 칠판 가득히 문제를 푸는 것처럼 화면전체에 걸쳐 문제가 나와 답을 맞추면 'RIGHT', 오답을 냈을때는 그 답을 친절히 가르쳐 주도록 프로그램을 작성하라.

```
1  REM ************************
2  REM * MULTIPLY THE NUMBERS *
3  REM ************************
5  HOME
7  VTAB 2: HTAB 9: PRINT "* MULTI
     PLY THE NUMBERS *"
10 N1 =  INT ( RND (1) * 100)
20 N2 =  INT ( RND (1) * 10)
30 N3 = N2 * N1
40  VTAB 9: HTAB 16: IF N1 < 10 THEN
      PRINT " ";
43  PRINT N1
45  HTAB 14: PRINT "X";
50  HTAB 16: IF N2 < 10 THEN  PRINT
     " ";
55  PRINT N2
60  HTAB 15: PRINT  CHR$ (95); CHR$
     (95); CHR$ (95)
80  INPUT "THE ANSWER IS ";AN$
90  VTAB 19: HTAB 13
100  IF  VAL (AN$) = N3 THEN  PRINT
     "RIGHT"
110  IF  VAL (AN$) <  > N3 THEN  PRINT
     "SORRY, ";N1;" X ";N2;" = ";
     N3
112  PRINT
115  PRINT
120  PRINT "PRESS M FOR MORE OR S
      TO STOP (M/S)? ";
130  GET A$
140  IF A$ = "S" OR A$ = "E" THEN
      END
150  GOTO 5

]RUN
        * MULTIPLY THE NUMBERS *
                85
              X  7
                ---
THE ANSWER IS 595
            RIGHT


PRESS M FOR MORE OR S TO STOP (M/S)?

]RUN
        * MULTIPLY THE NUMBERS *
                82
              X  8
                ---
THE ANSWER IS 646
           SORRY, 82 X 8 = 656
```

문제 110. RUN으로 수행시키면 LIST가 되어 첫번째 행부터 지워가는 프로그램을 작성하라.

```
10  REM
      HANDY-WIPE
11  REM
20 LO = 1:HI = 24:S = LO
30  LIST : LIST
40  FOR I = LO TO HI STEP S
50  INVERSE : VTAB I
60  PRINT  SPC( 40): NORMAL
70  VTAB I: CALL  - 868: NEXT
80 X = LO:LO = HI:HI = X
90 S = S *  - 1: GOTO 30
```

문제 111. INPUT되는 모든 문자중 "I"자 위에 ·가 찍히는 프로그램을 작성하라.

```
10  REM
     DAUGHTER OF EYES!
11  REM
20  HOME
30 II = 1
40 I1 = 11
50  VTAB I1: INPUT "INPUT:";I$
60 II$ = "INPUT:" + I$: FOR I = 1
      TO  LEN (II$):I1$ =  MID$ (
     II$,I,1)
70  VTAB I1: HTAB II: PRINT I1$: VTAB
     I1 + 1: HTAB II: PRINT  CHR$
     (32);: IF I1$ =  CHR$ (73) THEN
      VTAB I1 - 1: HTAB II: PRINT
      CHR$ (46);: VTAB I1 + 1
80 II =  PEEK (36) + 1:I1 = I1 +
     2 * (II = 1 AND I > 1): NEXT
```

문제 112. 다음의 전송문을 텔레타이프하는 프로그램을 작성하라. 단 문자 하나 하나가 나타나기 전에 **+>** 기호가 움직이도록 하고 문자 디스플레이 속도는 SPEED=210이고 문장의 아래 위는 「…」으로 막도록 한다. 또 7번째 줄에서 수행되도록 하라.

```
]RUN

----------------------------------------

ONE DAY,IN THE MIDDLE OF THE NIGHT,TWO D
EAD BOYS GOT UP TO FIGHT. BACK TO BACK,T
HEY FACED EACH OTHER; DRAW THEIR SWORDSA
ND SHOT EACH OTHER.A DEAF POLICEMAN HEAR
D THE NOISE,AND CAME AND KILLED THOSE TW
O DEAD BOYS.

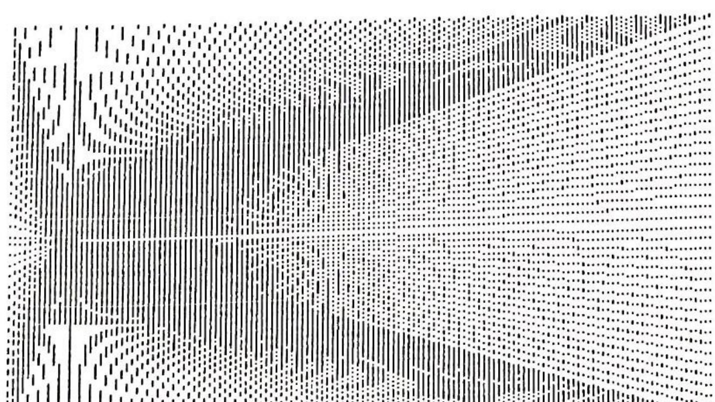
----------------------------------------
```

```
10  REM
       TELETYPER
11  REM
20 CUR$ = "+>": SPEED= 210
30  FOR I = 1 TO  LEN (CUR$):CUR$
     = CUR$ +  CHR$ (8): NEXT
40  HOME : VTAB 7: GOSUB 100
50 A$ = "ONE DAY,IN THE MIDDLE OF
      THE NIGHT,TWO DEAD BOYS GOT
      UP TO FIGHT. BACK TO BACK,T
     HEY FACED EACH OTHER; DRAW T
     HEIR SWORDSAND SHOT EACH OTH
     ER. ADEAF POLICEMAN HEARD TH
     E NOISE,AND CAME AND KILLED
     THOSE TWO DEAD BOYS.": GOSUB
     70
60  PRINT : GOSUB 100: SPEED= 255
     : VTAB 22: END
70  FOR I = 1 TO  LEN (A$): INVERSE
     : PRINT CUR$;: NORMAL :M$ =
     MID$ (A$,I,1): PRINT M$;: IF
    M$ = " " OR M$ = "." THEN  FOR
     J = 1 TO 25 + 150 * (M$ = ".
     "): NEXT
80  IF M$ <  > " " THEN  FOR J =
     1 TO 2:S =  PEEK ( - 16336):
     NEXT
90  NEXT : CALL  - 868: PRINT : RETURN
100 A$ = "": FOR I = 1 TO 40:A$ =
     A$ + "-": NEXT : GOSUB 70: RETURN
```

문제 113. 10진수의 0부터 65535까지를 16진수로 나타내는 프로그램을 작성하라.

```
10  REM
       ALL-HEX CONVERTER
11  REM
20 N = 16:S = N ^ 2:C = N ^ 3
30  HOME : FOR X = 0 TO 65535
40 A% = X / C
50 B% = (X - A% * C) / S
60 C% = (X - A% * C - B% * S) / N
70 D% = X -  INT (X / N) * N
80 A% = A% + 176 + 7 * (A% > 9)
90 B% = B% + 176 + 7 * (B% > 9)
100 C% = C% + 176 + 7 * (C% > 9)
110 D% = D% + 176 + 7 * (D% > 9)
120 T = T + 13: IF T = 13 OR T >
     27 THEN  PRINT :T = 1
130  HTAB T + (X < 10000) + (X <
     1000) + (X < 100) + (X < 10)
140  PRINT X;"=$"; CHR$ (A%); CHR$
     (B%); CHR$ (C%); CHR$ (D%);
150  NEXT
```

그래픽의 특성 및 활용



2－1 그래픽스의 기초

GR

```
]GR
```

물론 RETURN이 필요한 것은 기억하고 있을 것이다. 이 명령을 사용하면 텍스트의 위의 20행은 검게 클리어되고 프롬프트기호와 커서는 21행째로 옮긴다. 이것으로 저해상도 그래픽스 (Low Resolution Graphics ; 이하 LORES) 모드에 들어간 것이다. 따라서 아래 4 행은 텍스트를 위해서 남긴다.

TEXT

GR이란 그래픽스 (Graphics)에서 딴 것이다. 원래로 되돌아 가려면

```
TEXT
```

라는 명령을 실행하여라. 그렇게 되면 스크린은 돌연 @마크로 가득 채워지는데 그것으로 정상이다. TEXT커맨트를 실행하고 GR로 그래픽스로 되돌아가자.

색을 스크린에 내기전에 무슨색을 하고 싶은지를 컴퓨터에 일러두어야 한다. 다음과 같이 16가지의 색이 가능하다.

0 BLACK (흑색)
1 MAGENTA (약간의 푸른기가 있는 적색)
2 DARK BLUE (남색)
3 PURPLE (밝은 자색 혹은 등색)
4 DARK GREEN (어두운 녹색)
5 GREY (회색)
6 MEDIUM BLUE (청색)
7 LIGHT BLUE (밝은 청색)
8 BROWN (갈색)
9 ORANGE (주황색)
10 GREY (회색)
11 PINK (복숭아색)
12 GREEN (녹색)
13 YELLOW (노란색)
14 AQUA (물색)
15 WHITE (흰색)

이상과 같이 색에는 0부터 15까지 번호가 붙어있다.

노란색을 사용한다고 하자. 우선 GR커맨트를 실행하고 나서

COLOR=

```
]COLOR=13
```

이라고 타이프하라. 이것으로 앞으로 모든색이 어느것이나 노란색이다. 다음에 다른 색지정을 할 때까지 계속한다. 물론 아래 4 행의 텍스트부분은 제외다. 색을 스크린의 좌측위의 귀퉁이에 나타내려면

PLOT

```
]PLOT 0,0
```

이라고 타이프 한다.

컬러그래픽스를 스크린에 내는데는 16색중에서 내고싶은 색과 장소를 두어야함을 알았다. 장소의 지정은 스크린의 수직방향으로 40분할하고, 0부터 39까지 번호를 붙인다. 0의 컬럼은 스크린의 좌단이며 우측으로 감에 따라 번호는 증가한다.

마찬가지로 스크린은 수평방향으로 40분할되고 또 0부터39의 번호가 붙여진다 수평방향의 행은 스크린 상단 0부터 하단의 39까지 증가해 간다.

같은 색으로 우측위의 귀퉁이에 점을 나타내려면 수평방향39, 수직방향이 0이므로

```
]PLOT 39,0
```

이라고 타이프한다. 언제나 수평방향을 먼저 타이프해라. 그러면 오렌지색의 점을 좌측밑 귀퉁이에 나타내 보자.

우선 색을 변경한다.

```
]COLOR=9
```

스크린에는 아무것도 일어나지 않는다. 컴퓨터는 다음에 무엇이 PLOT될때까지 기억하고 있다. 자, 모퉁이에 점을 나타내 보자.

```
]PLOT 0,39
```

PLOT커맨드를 사용했을때, 잘 나오는 에러메세지는 2가지가 있다.

PLOT대신에

```
]PLAT
```

라든가

```
]PLOP
```

등으로 타이프했을 경우,

```
?SYNTAX ERROR
```

이고, 또 다른 에러메세지는 PLOT명령에 허용된 좌표의 범위를 넘어서 PLOT하려고 했을때, 즉

```
]PLOT 13,85
```

등으로 하면

```
?ILLEGAL QUANTITY ERROR
```

라고 나온다. 이 메세지는 PLOT범위를 넘어 스크린을 뛰어 나온 것을 뜻한다. PLOT문에서 허용되는 좌표는 수평방향39, 수직방향39까지이다. 다음과같이 수직방향에 39이상의 수를 지정하면

```
]PLOT 20,45
```

스크린 아래의 텍스트 부분에 이상한 캐릭터가 나타난다.
PLOT명령으로 음의 값을 사용하든지 하면, 역시

```
?ILLEGAL QUANTITY ERROR
```

의 메세지가 표시된다.

갈색의 수평선을 5에서 9까지, 수평방향10에서 그으려고 하면

```
]COLOR=8
]PLOT 5,10
]PLOT 6,10
]PLOT 7,10
]PLOT 8,10
]PLOT 9,10
```

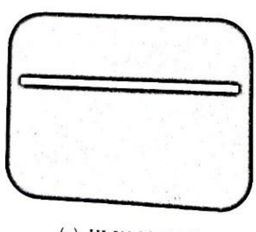
으로 하면 된다. 즉 서로 이웃해서 벽돌같은 점을 연결해가면 선처럼 보인다. 그러나 너무나 많은 시간을 소모한다. 예를 들면 좌측끝에서 우측끝까지 수평선을 그으려면, 선을 구성하기 위해 PLOT명령을 40회나 주어야 한다.

HLIN

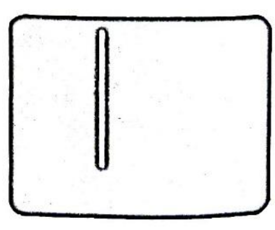
더 간단한 방법을 이용하자.

```
]GR
]COLOR=8
]HLIN 0,39 AT 10
```

RETURN 키를 누르면 순식간에 선을 구성한다. 이것은 수평선(Horizontal line)을 0부터 39까지 수직방향10의 위치에 긋는다는 것이다.



(a) HLIN(수평선)



(b) VLIN(수직선)

그림 2－1 수평선과 수직선

앞서의 점이나 선이 그어져 있는곳에 새로운 색을 넣으려면 먼저 색은 지워지고 새로 지정한 색으로 된다. 그래픽스 스크린을 모두 지울려면 GR명령을 사용하라.

수평선과 비슷한 수직선의 명령도 있다. 오렌지의 수직선을 수평방향 7 의 위치에서 12부터 33까지 그으려면 VLIN

```
]COLOR=9

]VLIN 12,33 AT 7
```

이라고 타이프한다.

수평, 수직의 수를 바꾸고 몇개의 수직선을 그어 보아라.

잘 되었는지를 알기위해 스크린 둘레를 COLOR=1을 이용해 둘러싸라. 이것이 된다면 녹색의 십자를 스크린에 그려라. COLOR=0 으로 하고 몇개의 선을 그어 보아라.

PLOT, HLIN, VLIN에 대해 어느정도 알때까지 시도해 보는 것이 좋다.

```
SCRN( X,Y)
```

는 GR모드에 있어서 X, Y 위치의 색을 알 수 있다. 다음의 프로그램의 예를 보자. SCRN(

```
* PROGRAM 2.1
 10 GR : COLOR= 9
 20 HLIN 0,39 AT 10
 30 PRINT SCRN( 10,10)
```

여기서 행번호 30의 SCRN(10, 10)은 좌표(10, 10)으로 지정되는 블록의 색코드를 수치의 값으로 한다. 따라서 9 라고 프린트한다.

```
]RUN
9
```

지금까지 설명한 것은 LORES였다. 그러면 LORES보다 훨씬 해상도가 뛰어나는 고해상도 그래픽스 (High Resolution Graphics ;이하 HIRES)에 대해 알아보자. HGR

이 새로운 그래픽스는 40×40의 저해상도 그래픽스보다 더욱 세밀하게 그릴 수 있다. HIRES는 280×160의 점을 나타낼 수 있다.

수평좌표는 좌단의 0에서 시작되어 우단의 279까지이고, 수직좌표는 상단의 0에서 하단 159에서 끝난다.

대부분의 HIRES용 명령은 머리에 "H"를 붙이는 것 외에는 LORES의 그것과 같다. 따라서 HIRES을 약칭하여 HGR로 표시한다.

HIRES모드를 세트하면 이 명령은 아래에 4 행의 텍스트 부분을 남기고 스크린 전면을 검게한다. HGR에서도 GR과 똑같이 아래의 4행의 텍스트도 좌표로서 생각할 수 있다. 따라서 이경우 수직좌표는 하단191까지가 되나 이들 점은 스크린에는 표시되지 않는다.

]HGR RETURN

커서가 보이지 않을때는 커서가 나타날때까지 RETURN 를 몇번 눌러라.

HCOLOR= HIRES는 참으로 훌륭한 것이나 이것을 사용하기 위해서는 몇개의 희생은 어쩔수 없다. 이 그래픽 모드에서 사용할 수 있는 색은 적어진다. 색의 번호는 0부터 7까지이다.

0 흑색 1	4 흑색 2
1 녹색	5 등색(가변)
2 자색	6 청색(가변)
3 백색 1	7 백색 2

이들 색은 TV에 따라서도 다르고 스크린상의 위치에 따라서도 다르다. 예를들면 색이 3번일때 나타나는 점의 색은 X축의 좌표가 짝수일 경우는 청색이되나 홀수일 경우는 녹색이 된다. 또 홀수와 짝수의 좌표가 PLOT되면 백색으로 나타난다.

이것은 가정용 TV의 탓이다.

HPLOT HIRES모드로 점을 그으려면 HPLOT로만 하면 된다. 다음과 같이 해보자 우선

```
]HGR
```

이라고 타이프하고 화면이 검게 되면서 아래 텍스트부분에 프롬프트와 커서가 나타나면

```
]HCOLOR=3

]HPLOT 130,100
```

X=130, Y=100의 좌표에 아주 작은 점이 나타날 것이다.
HIRES모드에서 선을 긋는 것은 LORES모드와 마찬가지로 간단하다.

HPLOT 시작점의 좌표 TO 끝나는 점의 좌표

라고 타이프하면 된다. 스크린 상단에 선을 그어보자.

```
]HPLOT 0,0 TO 279,0
```

우측 상단에서 우측 하단까지 선을 그으려면

```
]HPLOT TO 279,159
```

라고 타이프하면 우단에 선이 그어진다. 이 명령은 최후에 플롯한 점의 좌표를 시점으로 해서 같은색으로 선을 긋는다. HPLOT를 실행하기 전에 HCOLOR로 다른 색을 지정하면 선은 지정된 색으로 구성된다.

이와같이 이들 명령을 연결할 수 있으므로 1개의 스테이트먼트로 몇개의 선을 그을 수 있다.

```
]HPLOT 0,0 TO 279,0 TO 279,159 TO 0,159 TO 0,0
```

스크린 전체를 4각으로 둘러싸 보았다.

직선만이 아니고 사선도 간단히 그려진다.· 스크린의 좌측상단에서 우측하단까지 선을 그어보자.

```
]HPLOT 0,0 TO 279,159
```

여러가지 길이와 색으로 선을 그리자.

다음은 패들(paddle)을 이용해 HIRES모드로 스케치하는 프로그램이다. 패들이 준비되어 있으면 한번 시도해 보라. PDL

```
* PROGRAM 2.2
 10 HGR
 20 HCOLOR= 3
 30 X = PDL (0)
 40 Y = PDL (1)
 50 IF Y > 159 THEN Y = 159
 60 HPLOT X,Y
 70 GOTO 30
```

행50은 패들의 값이 0부터 255까지 있으나. Y좌표가 0 부터 159까지밖에 없으므로 Y의 값이 화면에서 뛰어나오지 않도록 하기 위한 것이다. 자, 프로그램을 RUN시켜보자.

이렇게 하여도 상관없지만 더 간단히 하도록 프로그램을 개량해 보자. 점과점의 갭은 반드시 필요한 것이 아니므로 행30이후를 다음 예와 같이 고쳐보면

```
* PROGRAM 2.3
 10 HGR
 20 HCOLOR= 3
 30 GOSUB 100
 40 HPLOT X,Y
 50 GOSUB 100
 60 HPLOT  TO X,Y
 70 GOTO 30
 100 X = PDL (0) / .913
 110 Y = PDL (1) / 1.6
 120 RETURN
```

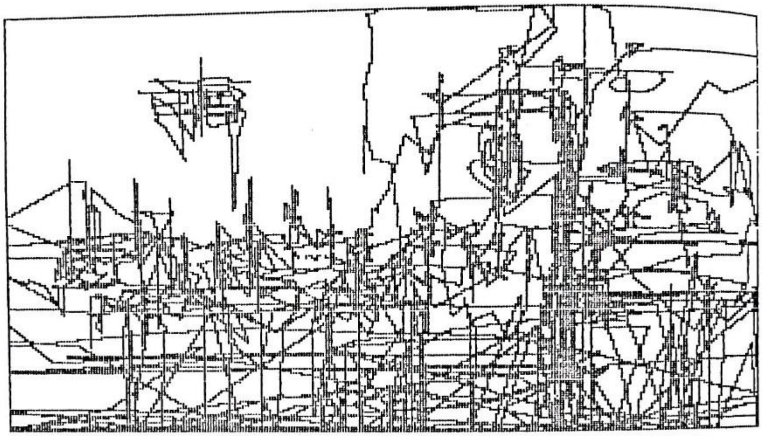
LORES모우드와 마찬가지로 주어진 값과 같거나 또는 그 이하의 정수에 대응한 좌표에 점이 찍힌다.

행120의RETURN으로 주 프로그램의 행40으로 되돌아와서 X, Y에 점을 치고 또 행50에서 서브루틴으로 뛰어서 X, Y의 값을 가지고 주 프로그램의 행60으로 돌아간다. 거기서 앞의 X, Y의 점부터 새로운 X, Y까지 짧은 선을 긋는다. 행70의 GOTO로 HGR과 HCOLOR을 제외한 프로그램을 반복한다. CTRL-C로서 프로그램을 정지한다.

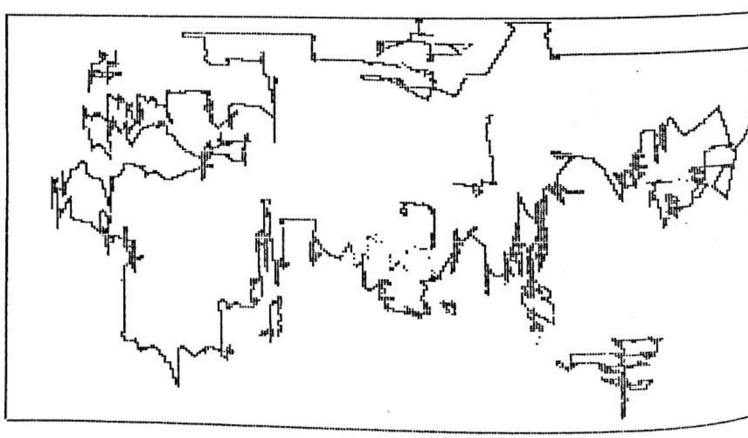
행100와 110에서 PDL(0)와 PDL(1)의 값을 나누는 것은 하나하나 점을쳐서 연결하는 대신, 선을 긋도록 하기 위해서이다. 게임패들의 최고값은 255이다. 따라서 스크린의 X좌표는 0부터 279까지 있으므로 0.913으로, 스크린의 Y좌표

는 최고 159이므로 1.6으로 나누는 것이다. 즉 점을 치는데는 어느정도 시간이 걸리고 점을 칠때에는 게임패들의 값의 변화에 대응할 수 없다.

따라서 개량하기 전의 프로그램으로 급히 움직이면 틈이 비게 되는 원인이다. 새로운 프로그램에 의해 해독한 새 좌표로 짧은 선을 그으면 이것을 피할 수 있다. 그 이유는 선을 그을 경우, 2점을 지정하면 그사이의 점은 자동적으로 플롯되어 한번에 1개의 점을 치는 것보다 빠르기 때문이다.



패들을 이용하여 그린 그림의 한 예



패들을 이용하여 그린 지도의 모양

2-2 그래픽 특성

애플Ⅱ로 모니터(monitor)에 표시하는 방법은 여러가지가 있지만 크게 다음의 3가지로 나눌 수 있다.

메모리의 할당과 그의 이용

①텍스트 모드(TEXT mode)

②저해상도 그래픽 모드(LO-RES mode)

③고해상도 그래픽 모드(HI-RES mode)

의 3종류이나 혼합 모드 그래픽을 할 수 있기 때문에 좀 더 자세히 알기 위해서 메모리의 할당과 이용에 관해서 이해가 요구된다. 실제에 있어서 그래픽 출력은 사실상 컴퓨터 메모리 일부의 내용을 화면에 출력하는 것이다.

사용자도 컴퓨터와 메모리 사이의 관계를 수정할 수 있으나 일반적으로 사용되는 구조는 그림2-2와 같이 메모리 번지들은 16진법과 10진법으로 표시된다.

메모리의 단위는 바이트(byte)인데 8개의 비트(bit)로 구성되어 있고, 각 비트는 1 또는 0이라는 정보를 기억하게 된다. 메모리내에서 각 바이트의 위치에 번지(address)를 정해 주는데, 하나의 번지와그 내용은 10진법, 2진법 또는 16진법으로 동일하게 나타낼 수 있다.

예를 들어 메모리 번지 4012에 64가 들어 있다면 이것을 표현하는 방법으로 다음과 같은 세가지 방법이 있다.

표 2-1 메모리번지 및 내용표현방식

	번 지	내 용
2진수표기법	0000111110101100	01000001
10진수표기법	4012	65
16진수표기법	0FAC	41

일반적으로 번지들을 나타내는데는 10진수 표기법이나 16진수 표기법을 사용하며 2진수 표기법은 컴퓨터내에서는 사용되지만 일반 사용자는 불편해서 사용되지 않는다.

이제부터 모든 메모리의 번지는 10진표기법과 16진표기법을 구별하기 위해서 16진법의 수 앞에는 「$」를 첨가하기로 하자.

애플은 48K RAM과 16K ROM을 갖고 있어서 메모리는 모두 64K 바이트로 이루어져 있다. 컴퓨터 분야에서는 1K가 2^{10}, 즉 1024를 나타내므로 48K는 실제로 49152 (=48×1024) 바이트이고 64K는 65536 (=64×1024) 바이트가 된다.

65536개의 메모리 번지는 10진수로 0에서부터 65535까지, 16진수로 $0000에서 부터 $FFFF 까지의 연속적인 숫자가 부여된다.

어떤 하나의 기억장소, 예로서 $1B82(7042)를 다루는데 있어서 $1B(27)를 상위바이트라 하고 $82(130) 를 하위바이트라고 부른다.

메모리 할당과 그의 이용

번지 16진수	번지 10진수	이용
$ FFFF	65535	
		시스템 이용 (system usage)
$ C000	49152	
		디스크 운영 체계 (disk operating system)
$ 9600	38400	
		문자열 (string)과 일반적 이용 (strings, general usage)
$ 6000	24576	
		고해상도 그래픽의 페이지 2 (page 2 of High Resolution Graphics)
$ 4000	16384	
		고해상도 그래픽의 페이지 1 (page 1 of High Resolution Graphics)
$ 2000	8192	
		범용 가변 배열 및 변수와 프로그램 (general usage variable array, variables, programs); $0BFF – $0800: 텍스트 페이지 2, 저해상도 그래픽 페이지 2 (TEXT page 2, Low Resolution Graphic page 2)
$ 0800	2048	
		텍스트 페이지 1, 저해상도 그래픽 페이지 1 (TEXT page 1, Low Resolution Graphic page 1)
$ 0400	1024	
		시스템이용 (system usage)
$ 0000	0	

그림 2 – 2 메모리 할당과 그의 이용

메모리 번지 ⇨ $ [1B] [82]

↑ 상위바이트　↑ 하위바이트

각 모드가 정보를 그리는데 2개의 영역이 있다.

최초의 영역은「제 1 페이지」또는「페이지 1」이라고 한다. 2번째 영역은「제 2 페이지」또는「페이지 2」라고 부르고 최초의 영역의 바로 다음에 있어서 크기가 같다. 즉시 표시하고 싶은 그림어나 문자를 넣어 두는데 사용하면 편리하다.

표 2-2 화면표시 메모리 범위

화 면	페이지	시 작 번 지	최 종 번 지
텍스트/저해상도 그래픽	1	$0400 (1024)	$07FF (2047)
	2	$0800 (2048)	$0BFF (3071)
고해상도 그래픽	1	$2000 (8192)	$3FFF (16383)
	2	$4000 (16384)	$5FFF (24575)

메모리 번지 $0400 (1024) 부터 $07FF (2047) 까지는 텍스트 모드와 저해상도 그래픽을 위해서 사용된다. 만약 텍스트 모드로 사용되면 컴퓨터는 이 메모리 번지들의 내용을 텍스트로 받아들여 처리해 주며 또 이것은 24행씩 화면에 나타내 준다.

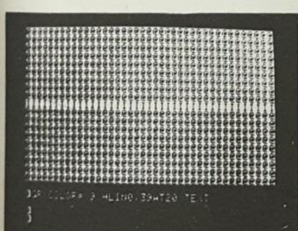


그림 2-3 TEXT

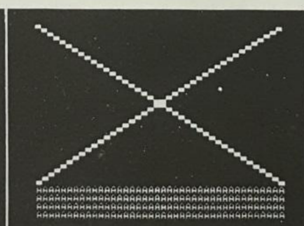


그림 2-4 LO-RES MIXED

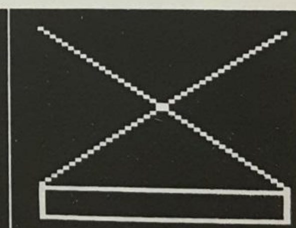


그림 2-5 LO-RES FULL

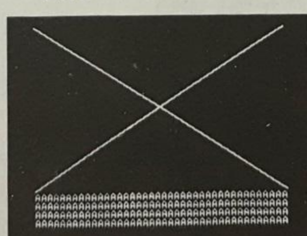


그림 2-6 HI-RES MIXED

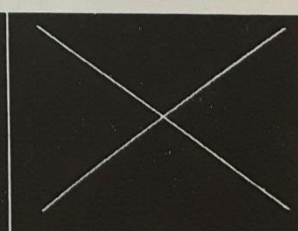


그림 2-7 HI-RES FULL

저해상도 그래픽 모드에 있을 때는 컴퓨터는 이 메모리 번지의 내용을 그래픽과 텍스트로 해석하여 화면의 아래부분에 4행의 텍스트와 40×40의 그래픽 배열로서 그 윗 부분에 나타낸다. 메모리 번지 $0800 (2048) 부터 $0BFF (3071) 까지는 텍스트 또는 그래픽스 표현을 위해서 사용될 수도 있는데 일반적으로 애플 소프트 BASIC 프로그램들이 이 곳에 기억된다.

메모리 번지 $2000 (8192) 부터 $3FFF (16383) 까지는 보통 고해상도 그래픽스 제 1 페이지를 위해서 사용된다. HGR모드에 있을 때 컴퓨터는 이 메모리 지역에 있는 내용을 그래픽스로 해석하여 화면에 280×160 그래픽 배열과 그 아래 부분에 텍스트 4행으로서 나타난다. 이 텍스트들은 텍스트 페이지 1 ($0400~$07FF)의 마지막 4행에 해당된다.

고해상도 그래픽 페이지 2는 메모리 번지 $4000 (16384) 에서 부터 $5FFF (24575) 까지가 이용된다. HGR 2 모드에 있을 때 컴퓨터는 메모리의 이 지역을 텍스트가 없는 280×192 그래픽 배열로 화면에 나타난다.

고해상도 그래픽 페이지 2를 화면 아래쪽에 텍스트와 함께 나타낼 수도 있는데, 이 때 텍스트는 텍스트 페이지 2에서의 마지막 4행이다. 또한 고해상도 그래픽 페이지 1을 화면 아래쪽에 텍스트없이 280×192의 그래픽 배열로서 나타낼 수도 있다. 마찬가지로 저해상도 그래픽 페이지 1과 2도 각각 텍스트 없이 디스플레이 할 수도 있다.

화면출력을 위한 메모리번지 지정법

메모리 번지 $0400 (1024) 부터 $07FF (2047) 까지는 텍스트 디스플레이와 저해상도 그래픽 페이지 1로 사용되는데 그림2-8에 있는 문자는 다음과 같은 두가지 형태의 명령어로 나타낼 수 있다.

```
* PROGRAM  2.4

  10  TEXT : HOME : VTAB 6: PRINT  TAB(
       18);"TEXT"
  20  PRINT  TAB( 15);"ADDRESSING"
```

또는

```
* PROGRAM  2.5

  10  TEXT : HOME
  20  FOR ADDR = 1681 TO 1684
  30  READ X: POKE ADDR,X
  40  NEXT ADDR
  50  FOR ADDR = 1806 TO 1815
  60  READ X: POKE ADDR,X
  70  NEXT ADDR
  80  DATA  212,197,216,212
  90  DATA  193,196,196,210,197,211
       ,211,201,206,199
```

화면 위에서의 특정한 문자 위치에 해당되는 메모리 번지는 행번지와 열번지코드를 더하여 사용된다. 예를 들면 그림 2-8에서 문자 「R」을 갖고 있는 화면의

블록의 번지를 지정하기 위해서는 행번지 1664에 열번지 11을 더하여 대응되는 메모리번지 1675를 얻는다.

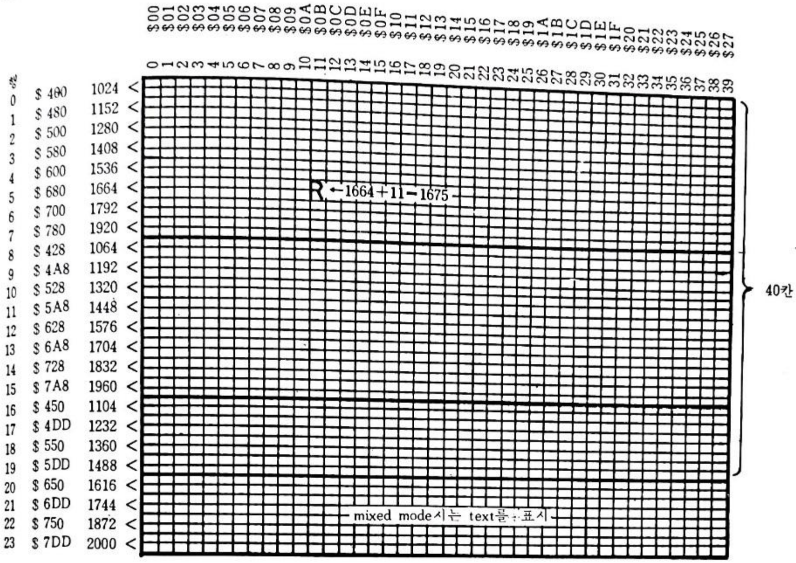


그림 2 − 8 저해상도 그래픽 화면의 메모리 맵(memory map)

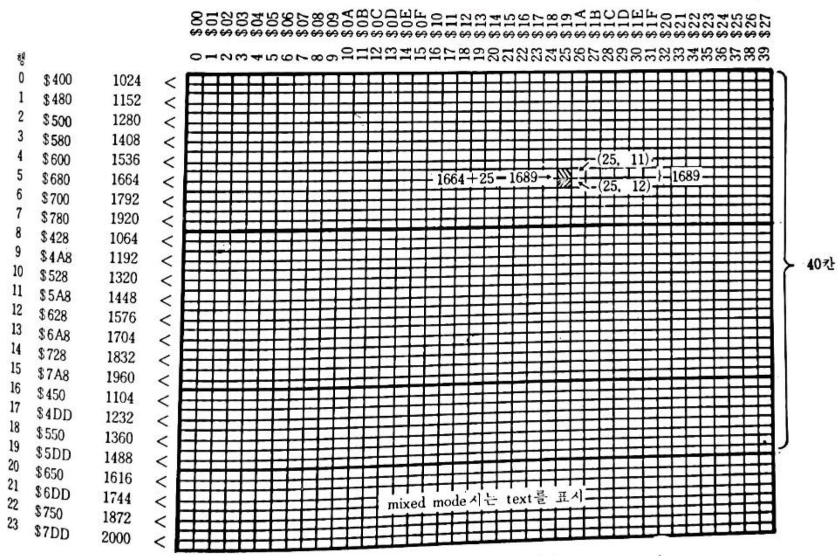


그림 2 − 9 저해상도 그래픽 화면의 메모리 맵(memory map)

그림 2-9의 빗금친 블록의 번지를 지정하기 위해서는 행번지 1664에 열번지25를 더하여 1689를 얻는데, 이 번지는 화면상에서 40×40=1600개의 블록 중에서 2개의 블록을 제어한다.

즉 화면에서의 (25, 11) 블록의 색은 번지 1689의 하위 4 비트의 내용에 의해서 결정되고 (25, 12) 블록의 색은 번지 1689의 상위 4 비트의 내용에 의해서 결정된다.

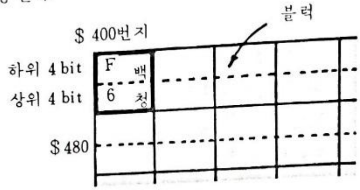


그림 2-10 LORES 메모리내의 데이타와 해당 블럭간의 관계

그림 2-11은 고해상도 그래픽 페이지 1에서 번지가 할당되는 방식을 보여준다. 고해상도 그래픽 페이지 2에 관한 것은 그림 2-9에서와 비슷한 방식으로 수행된다.

화면에서의 각 블록은 8개의 바이트와 관계가 있는데 그림 2-11에서와 같이 어느 한 블록의 행번지와 열번지의 합은 그 블록과 관련된 맨 위의 바이트에 대한 번지이고 나머지 7개의 바이트의 번지는 메모리 번지들을 연속해서 1024씩을 더함으로써 얻을 수 있다.

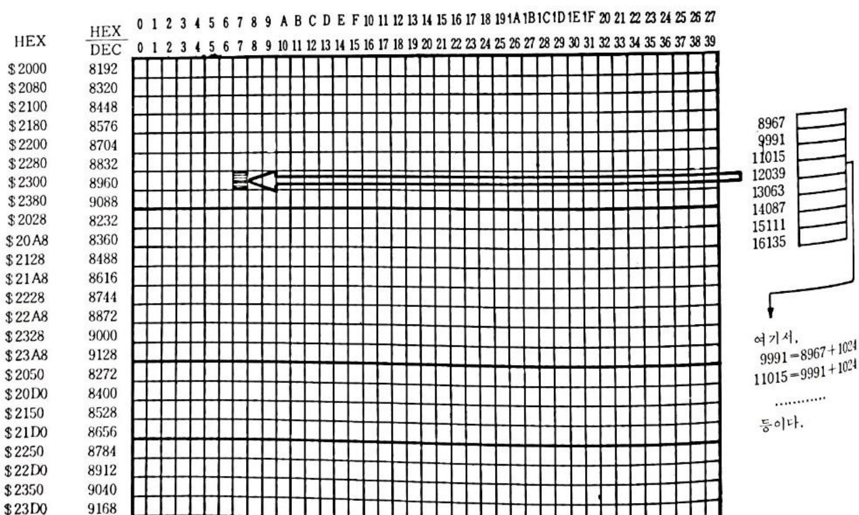


그림 2-11 고해상도 그래픽 페이지 1

TEXT 모드

텍스트 모드에서는 애플은 각행 40자로 24행의 문자를 표시한다. 화면상의 각 문자는 표시되어 있는 페이지의 메모리 영역의 한개의 메모리 번지의 내용을 나타내고 있다. 문자 세트는 26개의 대문자, 10개의 수, 28개의 특수문자로 전부 64문자이다.

문자는 폭 5도트, 높이 7도트의 도트 메트릭스로 만들어져 있다. 이웃한 문자를 분리하기 위해 각 문자 양쪽에 1도트폭의 스페이스가 있고, 상하의 행을 분리하는데는 각 행의 위의 1도트 높이의 스페이스가 있다. 문자는 보통 검은 바탕에 흰점으로 만들어지지만 또 흰바탕에 검은 점이든지, 흑백을 바꾸어서 반전하는 문자도 가능하다.

제1텍스트 페이지에 사용하는 메모리 영역은 1024번지에서 시작하여 2047번지에서 끝난다. 제 2화면은 2048번지에서 3071번지 까지이다. 기계어에서는 제 1페이지는 $0400에서 $07FF까지이고, 제 2페이지는 $0800에서 $0BFF까지이다. 각 페이지는 1024바이트의 길이이며, 단순한 곱셈을 하면 화면상에 표시되는 것은 960문자 뿐인 것을 알 수 있다. 각 페이지의 나머지 64바이트는 화면에 표시되지 않고 애플 인텔리전트 인터페이스 R주변 보드의 PROM에 스토어된 프로그램이 일시적인 기억 장소로서 사용하고 있다.

화면의 번지 지정 방법은 애플소프트 BASIC 언어 명령어인 POKE를 이용하여 메모리의 특정한 위치에 숫자를 저장할 수 있다.

예를 들면 POKE 1390, 217은 메모리 번지 1390에 숫자 217을 기억 시키라는 명령이다. 만약에 컴퓨터가 텍스트 모드로 동작되고 있었다면, 이 명령에 의해 문자 Y를 19번째 줄의 31번째 위치에 배치하는 것이 된다. 왜냐하면 문자 Y에 대한 ASCII 화면문자 코드가 10진수로 217이기 때문이다.

반전된 문자(inversed) Y를 윗줄에 디스플레이 하려면 POKE 1262, 25를 사용하면 되고, 깜박이는 문자(flashing) Y를 정상적인 문자(normal) Y의 아래에 놓으려면 POKE 1518, 89를 사용한다.

만약 위의 명령들이 하나씩 따로따로 순차적으로 수행되면, 화면이 위로 한 줄씩 밀려 올라가는 효과(scrolling)에 의해 위에서와 같은 효과가 나오지 않을 수도 있다. 이것은 다음과 같이 함으로써 해결될 수 있다.

```
]TEXT:HOME:POKE 1390,217:POKE 1262,25:POKE 1518,89
```

표 2-3에는 64문자의 정상적인 문자, 반전된 문자, 깜박이는 문자 표시 모드에 대응하는 10진, 16진 코드를 표시했다.

저해상도 그래픽 (LO-RES) 모드

저해상도 그래픽 모드에서는 텍스트 모드와 같은 1024바이트의 메모리의 내용을 다른 형식으로 표시한다. 이 모드에서는 메모리의 각 바이트는 ASCII문자로서 표시되는 것이 아니고, 1개의 블록 위에 또 1개 포개진 2개의 블록으로서 표시된다. 화면은 폭이 40블록, 높이가 48블록의 배열을 표시한다. 따라서 40×40의 그래픽 배열과 그 아래에 4행의 텍스트가 나타나게 되는데 한개의 메모리

표 2 – 3 ASCII 화면 문자

		Inverse				Flashing				(Control)		Normal				(Lower-case)	
Decimal		0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
	Hex	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
0	$0	@	P		0	@	P		0	@	P		0	@	P		0
1	$1	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	!	1
2	$2	B	R	″	2	B	R	″	2	B	R	″	2	B	R	″	2
3	$3	C	S	#	3	C	S	#	3	C	S	#	3	C	S	#	3
4	$4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	$	4
5	$5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	%	5
6	$6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	&	6
7	$7	G	W	'	7	G	W	'	7	G	W	'	7	G	W	'	7
8	$8	H	X	(	8	H	X	(	8	H	X	(	8	H	X	(	8
9	$9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	)	9
10	$A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	*	:
11	$B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	+	;
12	$C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	,	<
13	$D	M	]	–	=	M	]	–	=	M	]	–	=	M	]	–	=
14	$E	N	∧	.	>	N	∧	.	>	N	∧	.	>	N	∧	.	>
15	$F	O	·	/	?	O	–	/	?	O	–	/	?	O	–	/	?

번지의 내용은 2 개의 화면 블록의 색을 제어하는데 사용된다.

즉 저해상도 그래픽의 페이지 중의 각 비트는 세로로 겹친 2 개의 블록을 나타내므로, 각 바이트는 2 개의 같은 부분으로 분할된다. **니블** (nibble) 이라고 불리는 각 구분은 0 부터 15까지의 수치를 갖고 있다. 바이트 아래의 니블에 들어가 있는 수치는 화면상의 2 바이트 위의 블록의 색을 정하고, 위의 니블에 들어 있는 수치는 화면상의 2 바이트 아래의 블록의 색을 정한다. 색은 0 에서 15까지의 번호가 붙여져 있다.

니블 (nibble)

원래 한 입이라든가 조금씩 물어뜯는다는 의미로 사용되는 것으로 작은 단위를 가리키고 있는데 컴퓨터 분야에서는 8 비트를 바이트(byte), 16비트를 하프워드(half word), 32비트를 워드(word)라 하는데 비해 4 비트를 니블이라 한다.

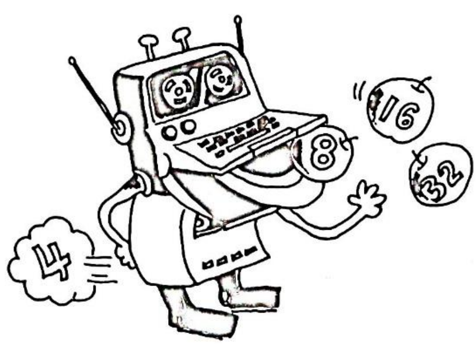


표 2 - 4 저해상도 그래픽의 색

10진	16진	색	10진	16진	색
0	$ 0	흑	8	$ 8	갈
1	$ 1	심홍	9	$ 9	오렌지
2	$ 2	암청	10	$ A	회 2
3	$ 3	적자	11	$ B	분홍
4	$ 4	암록	12	$ C	명록
5	$ 5	회 1	13	$ D	황
6	$ 6	청	14	$ E	남록
7	$ 7	명청	15	$ F	백

따라서 16진수 $ D8 이 들어 있는 바이트는 화면상에는 황색의 블록위에 갈색의 블록으로서 표시된다. 10진의 경우에는, 2 바이트의 수치를 16으로 나눈 몫이 아래 쪽의 블록의 색을 정하고, 나머지가 윗쪽의 블록의 색을 정한다.

다음의 예를 보자.

10진으로 25는 $ 19이므로 POKE 1390, 25는 상위 블록을 오렌지색(색코드9)으로, 하위 블록을 심홍색(색코드 1)으로 만든다. 이때 상하위 블록은 1390에 의해 각각 블록 (30, 36)과 (30, 37)을 가리키게 된다.

그럼 POKE 1262, 89를 생각해 보자.

여기에서 10진 89는 $ 59이고 1262는 두개의 블록(30, 34)와 (30, 35)를 지정하게 된다. 그러므로 POKE 1262, 89는 블록(30, 34)를 오렌지색으로, 블록(30, 35)는 회색으로 만들게 된다. 마찬가지로 POKE 1518, 217은 블록(30, 38)을 오렌지색으로 하고 블록(30, 39)를 노란색으로 만들 것이다.

고해상도 그래픽 (HI-RES) 모드

만약에 컴퓨터가 HGR 또는 HGR 2 모드에 있다면 화면에 나타나는 모양은 관련된 컴퓨터 메모리 내용의 각 비트에 대응하여 출력된다.

HGR모드일 경우, 화면의 맨위는 메모리 번지 $ 2000(8192)에 의해서 제어된다. 사실상 메모리 번지 $ 2000(8192)의 처음의 7개 비트가 HGR출력시 화면의 맨윗줄 처음 7개점에 대응되어 나타나고 8번째 비트는 나타나지 않고 색을 제어하는데 이용된다.

예를 들어 메모리 번지 8192가 숫자 46을 갖고 있다면 46의 2진수 표현이 0101110이다. 그래픽 디스플레이는 각 비트들이 역순으로 나타나게 되므로 그림 2-12에서와 같이 2,3,4,6번째 점은 ON되어 켜지고 나머지는 모두 OFF되어 꺼지게 된다.

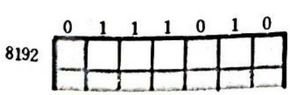


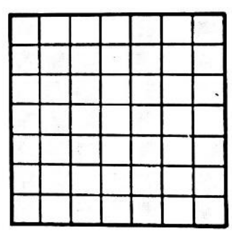
그림 2 - 12 POKE 8192, 46

다음의 프로그램은 각 데이타가 비트별로 화면에 나타나는 것을 이해하기 위한 것이다. 이 프로그램을 실행시켜 보자.

```
* PROGRAM  2.6
 10  DATA  46,40,40,104,40,36,34
 20  HOME : HGR : VTAB 21
 30  FOR I = 8192 TO 14366 STEP 10
      24
 40  READ X
 50  POKE I,X
 60  INPUT K$
 70  NEXT I
 80  END
```

행번호 30에서 FOR문은 메모리 번지 8192, 9216, 10240, 11264, 12288, 13312, 14366과 관계가 있으며 이 메모리 번지에 있는 내용들은 행 60에 의해 RETURN 키를 누를 때마다 화면의 좌측 상단에 하나씩 하나씩 디스플레이 된다.

이 7개의 메모리 번지 가운데 있는 내용을 화면에 나타내면 화면 좌측 상단구석에 문자「가」가 그려진다.



메모리 번지	2진법	10진법
8192	0101110	46
9216	0101000	40
10240	0101000	40
11264	1101000	104
12288	0101000	40
13312	0100100	36
14366	0100010	34

그림 2-13 비트패턴의 문자 디스플레이

우리는 위와 같은 방법에 의하여 그래픽 화면에 여러가지 텍스트들을 쉽게 나타낼 수 있게 되었다. 여러가지 문자들을 실제로 구성해 보라.

각 문자 부호들은 7×7 배열 내에 나타낼 수 있다. 구성절차는 각 비트 패턴들을 숫자코드로 변형시키고 다음에 이 비트패턴들을 POKE 명령을 사용하여 기억장소에 기억시키면 된다.

색(COLORS)

컬러 그래픽 출력에 있어서 제약사항이 있다. 이 제약사항은 컬러를 다루는데 있어서 중요한 문제점이다.

즉 화면상의 모든 블록에 제공된 모든 색으로 그릴수가 없는데, 이것은 다음의 프로그램을 수행해 보면 알 수 있을 것이다.

* PROGRAM 2.7

```
0  REM  PLOTTED LINES ARE NOT VIS
     IBLE
10  HOME : HGR : VTAB 21
20  HCOLOR= 2
30  FOR I = 2 TO 5
40  HPLOT 7 * I,10 TO 7 * I,30
50  NEXT I
60  END
```



위의 프로그램은 4개의 수직선을 그리도록 하였지만 실제 2개의 선만이 화면에 나타난다. 이것은 HCOLOR＝2 (청색)은 짝수째 열에서만 나타날 수 있기 때문이다.

일반적으로 HCOLOR 1과 5는 홀수번째 열에서만, HCOLOR 2와 6은 짝수번째 열에서만 나타날 수 있다. 그러나 흰색과 검은 색은 모든 열에서 가능하다.

또 한가지 컬러에 있어서의 문제점은

* PROGRAM 2.8

```
0  REM  PLOTTING OVER BACKGROUND
     COLOR
10  HOME : VTAB 21: HGR : HCOLOR=
     2
20  HPLOT 0,0
30  CALL 62454
40  HCOLOR= 5
50  HPLOT 1,1 TO 100,100
60  END
```

와 같이 행 20, 30에서 화면의 바탕색은 행 10에 의해 청색이며, 이때 행 50에서 HCOLOR＝5로 대각선을 그리면 그림 2-14와 같이 선이 굵직 굵직하게 뭉쳐서 연결되어 보기가 좋지 않는, 혼합색에 대한 문제점이 있다.

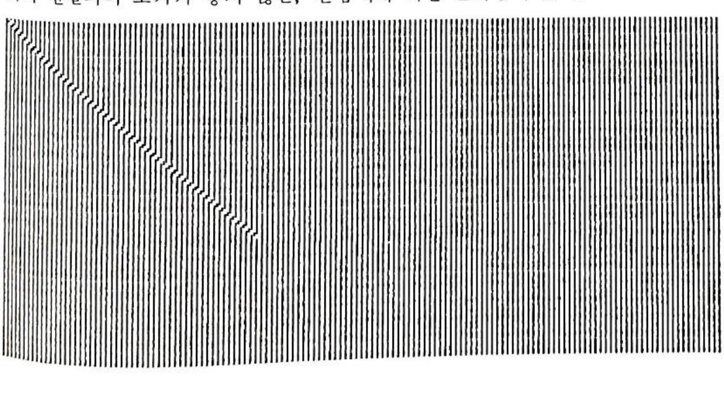


그림 2 - 14

해상도

해상도(resolution)라는 용어를 그래픽 화면에 적용할 때는 그래픽스 하는데 사용되는 블록 또는 화소(pixel)의 갯수를 의미한다.

HPLOT X, Y라는 명령에서 X는 0에서 279까지의 정수를 받아들이고 Y는 0에서 191까지의 정수를 받아들인다. 이러한 이유로 고해상도 그래픽은 53760개의 280×192의 블록 또는 화소를 제공하는 해상능력을 갖는다.

그러나 화면의 모든 블록을 허용된 모든 색으로 나타낼 수 없기 때문에

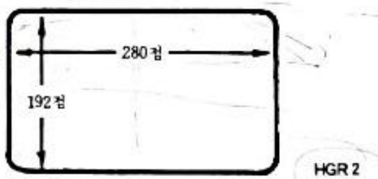
```
]HGR:HCOLOR=1:HPLOT 20,10 TO 20,100
```

따라서 실제에 있어서의 컬러 그래픽스 해상도는 280×192가 아니라 140×192라고 할 수 있다.

```
]HGR:HCOLOR=3:HPLOT 10,10 TO 100,100
```

을 실행시키면 의도했던 바와 같이 한 점의 두께를 갖는 선이 아님을 알 수 있다.

```
]HCOLOR=0:HPLOT 9,10 TO 99,100
```



앞에서의 선이 필요 이상으로 누껍다는 것을 알았을 것이다.

보다 높은 해상도를 얻기 위해

```
* PROGRAM  2.9
 10  HOME : HGR
 20  FOR I = 1 TO 15
 30  READ X
 40  POKE 14900,X
 50  VTAB 22: PRINT I
 60  FOR J = 1 TO 500: NEXT J
 70  NEXT I
 80  DATA  1,129,2,130,4,132,8,136
     ,16,144,32,160,64,192,128

 90  END
```

프로그램이 실행될 때 자세히 살펴보면 한 바이트의 내용을 변경시킴으로써 15개의 점이 순차적으로 디스플레이 되고 그 카운터가 22행에 나타남을 알 수 있다. 그런데 그래픽 화면은 40바이트를 그릴 수 있는 폭을 갖고 있으므로 여기에서 560×192의 해상도를 얻을 수 있다.

직선 (LINE)

이론적으로 직선은 폭이나 두께의 크기가 0이나 화면상에서의 직선의 도해적

인 표현법으로는 직선을 구성하는 화소가 나타나거나 그렇지 않을 경우, 화소의 크기가 점이 아니기 때문에 직선의 표현은 두께를 갖는다. 더우기 직선을 나타내기 위해서 필요한 화소가 행과 열에 직선과 유사하게 배열되므로 대각선을 그릴 때는 그 결과가 직선이 될 수 없다.

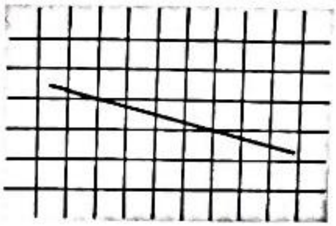


그림 2-15 직 선

그림2-15와 같은 선을 생각해 보자. 만약 선이 통과하는 모든 화소들을 ON 시키면 선의 화상은 계단 모양을 갖게 된다.

직선이 정의되어지는 방법도 직선의 형태에 영향을 준다. 직선을 그릴 때 애플소프트 BASIC은 어느 화소가 나타나야 할 것인가를 결정하는데 직선이 정의되는 방향이 중요한 역할을 한다.

```
]HGR:HCOLOR=3:HPLOT 10,10 TO 100,99
```

위의 명령문으로 대각선을 그리고 나서 화면을 지우지 않고

```
]HPLOT 100,99 TO 10,10
```

주의깊게 살펴보면 비록 동일한 직선을 요구하였지만 그 형태는 약간 다르다. 이것은 방향이 반대이기 때문이다.

```
]HGR:HCOLOR=3:HPLOT 10,10 TO 100,90

]HCOLOR=0:HPLOT 100,90 TO 10,10
```

검은 색의 선을 그음으로써 흰 선을 지워버려야 하지만 약간의 자국이 남아 있다. 직선의 방향에 의해 야기되는 차이는 그다지 중요한 것은 아니나, 때때로 만족할 만한 형태를 얻기 위해 세심한 주의가 필요하다.

HCOLOR 지정

그림을 그리는 명령문을 사용하기 전에 그림을 그리는데 필요한 메모리의 영역, 즉 HGR 또는 HGR 2와 사용할 색을 지정하는 것이 필요하다.

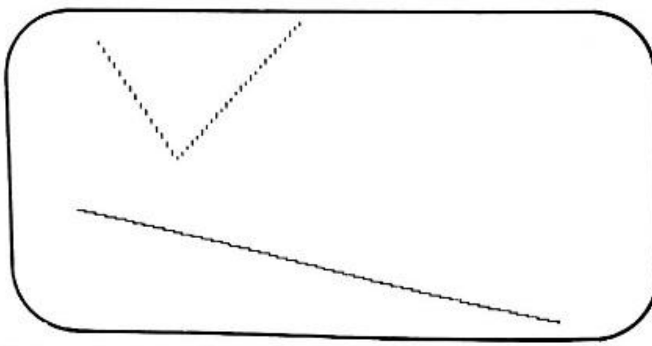
일단 HCOLOR가 지정되면 그것은 다음의 「HCOLOR=」문을 만날때까지 H-PLOT와 HPLOT TO에 있어서 고정된 상태로 남는다. 그러나 HCOLOR가 새로운 색으로 지정될 때마다 HPLOT TO가 아니라 HPLOT 명령문이 뒤따라야 한다.

그 이유는 HPLOT TO X, Y가 수행될 때는 그림을 그리기 위해 가장 나중에

사용된 색이 나타나는 반면에, HPLOT X, Y는 항상 가장 나중에 「HCOLOR=」문에 의해서 지정된 색을 사용하기 때문이다.

몇개의 HPLOT TO X, Y문 사이에서 HCOLOR를 바꾸고 싶을 때 HPLOT X, Y문을 넣지 않고 색을 바꿀 수 있으면 편리한데 다음과 같이 하면 된다. 메모리 번지 $ 00E4 (228)은 가장 나중에 지정된 HCOLOR의 컬러 코드를 가지고 있고 메모리 번지 $ 001C (28)은 가장 나중에 그림을 그리는데 사용한 HCOLOR의 컬러 코드를 가지고 있다. 그러므로 만약에 우리가 메모리 번지 $ 001C (28)의 내용을 POKE문을 사용하여 바꾼다면 우리는 애플소프트 BASIC이 모르게 색을 바꿀 수가 있다.

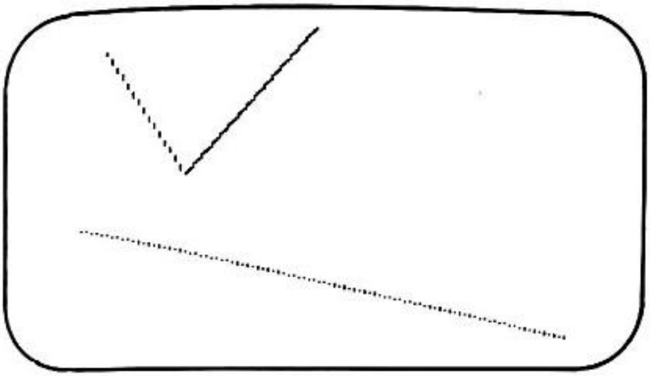
```
* PROGRAM  2.10
 10  HOME : VTAB 21: HGR : HCOLOR=
      2
 20  HPLOT 20,20 TO 50,75
 30  HCOLOR= 3
 40  HPLOT  TO 100,10
 50  HPLOT 10,100 TO 200,150
 60  END
```



위 프로그램에서 행 번호 30에서 새로 지정된 색은 행 40에서는 효력이 없고 오직 행 50부터 효력이 발생된다. 위 프로그램에서 행번호 30을 다음과 같이 변경하자.

```
30  POKE 28,127: REM  COLOR CODE
     OF HCOLOR=3
```

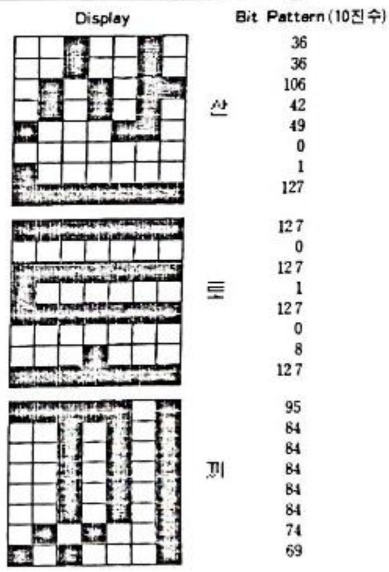
변경된 프로그램의 행 30에서의 색의 변화는 행 40에 영향을 주는데 애플소프트 BASIC은 가장 나중에 사용된 HCOLOR가 3이라고 생각하기 때문이다. 행 50에서는 가장 나중에 HCOLOR문에 의해 행 10에서 지정된 색(코드2)으로 남아 있어 푸른색으로 그리게 된다.



문제 114. 문자 "산토끼"를 화면의 상단에 다음과 같이 디스플레이 되도록 각 비트 패턴(bit pattern)들을 숫자코드로 변형시켜 프로그램을 작성하라.

산 토 끼

```
10  HOME : HGR : VTAB 21
20  DATA  36,36,106,42,49,0,1,127
30  FOR I = 8205 TO 15373 STEP 10
     24
40  READ Y
50  POKE I,Y
60  NEXT I
70  DATA  127,0,127,1,127,0,8,127
     : FOR I = 8209 TO 15377 STEP
     1024: READ Y: POKE I,Y: NEXT I
80  DATA  95,84,84,84,84,84,74,69
     : FOR I = 8213 TO 15381 STEP
     1024: READ Y: POKE I,Y: NEXT I
90  END
```





2-3 그래픽모드의 응용

간단한 명령어들을 키보드에서 타이프함으로써 화면에 그림을 그릴 수 있다.



저해상도 그래픽 모드를 이용한 윙크하는 얼굴 그리기

```
]GR
```
RETURN

은 컴퓨터를 저해상도 그래픽 모드로 만드는데 이것은 X, Y방향으로 0부터 39까지 번호가 할당된 40×40의 블록들로써 화면을 나누어 취급한다.

```
]COLOR=15
```

는 일반적인 모니터를 사용할 때와 또 애플이 컬러 디스플레이 장치에 연결되어 있을 때 그림이 백색으로 나타나게 만든다. 다음 명령어

```
]HLIN 10,30 AT 5
```

는 얼굴을 구성하는 맨 위 수평선을 그린다. HLIN을 구성하는 문자는 화면에서 수평선을 그리는데 이 명령어 AT 앞에서는 선분의 시작과 끝점의 X좌표를 지정하며 AT다음에는 이에 대응되는 Y좌표를 나타낸다. 그 다음의 두개의 명령어

```
]HLIN 15,16 AT 14

]HLIN 23,24 AT 14
```

는 왼쪽과 오른쪽 눈을 그린다. 코는 아래의 명령어들에 의해서 그려진다.

```
]VLIN 16,18 AT 19

]VLIN 16,18 AT 20
```

VLIN을 구성하는 문자는 수직선을 의미하는데 AT앞에 선분의 시작과 끝점의 Y좌표를 지정하고 AT뒤에는 그 선분의 X좌표를 나타낸다. 입, 얼굴을 구성하는 좌우의 선과 아래에 있는 선은 다음과 같은 명령들에 의해서 그려진다.

```
]HLIN 15,24 AT 24

]VLIN 5,30 AT 10

]VLIN 5,30 AT 30

]HLIN 10,30 AT 30
```



그림 2 - 16

다음의 명령어를 첨가하여 입 양끝에 블록 하나씩을 덧붙임으로써 그림 2-17에 나타난 것과 같은 미소짓는 얼굴을 만들 수 있다.

```
]PLOT 15,23

]PLOT 24,23
```

여기서 PLOT X, Y는 화면상의 좌표(X, Y)에 점을 그리는 명령어이다.

이제부터 화면상에서 그림의 움직임을 고려해 보자.

미소짓는 얼굴에서의 두 개의 눈이 그림 2-17에 나타나 있는데, 각각의 눈은 저해상도 그래픽 모드에서 화면을 구성하는 40×40개의 블록 중에서 9개를 차지하고 있다.

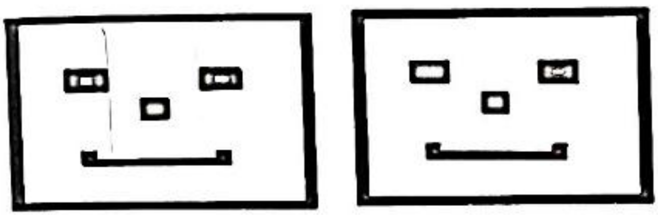


그림 2-17

다음의 프로그램 「WINKFACE」는 얼굴을 화면에 그리고 왼쪽눈을 교대로 떴다 감았다 하는 프로그램이다.

* PROGRAM 2.11

```
10  REM  ***********************
20  REM  * W I N K F A C E    *
30  REM  ***********************
40  REM
50  GR : COLOR= 15
60  REM  DRAW FACE WITH OPEN EYES
70  HLIN 10,30 AT 5: HLIN 14,16 AT
     14: HLIN 23,25 AT 14
80  VLIN 16,18 AT 19: VLIN 16,18 AT
     20: HLIN 15,24 AT 24
90  VLIN 5,30 AT 10: VLIN 5,30 AT
     30: HLIN 10,30 AT 30
100 PLOT 15,23: PLOT 24,23
110 HLIN 14,16 AT 12: HLIN 23,25
     AT 12
120 PLOT 14,13: PLOT 16,13
130 PLOT 23,13: PLOT 25,13
140 REM  CLOSE LEFT EYE
150 COLOR= 15: PLOT 15,13
160 REM  TIME DELAY WHILE IT IS
     CLOSE
170 FOR DE = 1 TO 100: NEXT DE
180 REM  OPEN LEFT EYE
190 COLOR= 0: PLOT 15,13
200 REM  TIME DELAY WHILE IT IS
     OPEN
210 FOR DE = 1 TO 100: NEXT DE
300 GOTO 150
```

행번호 70~130은 이미 설명한 저해상도 그래픽의 명령어를 이용하여 두 눈을 뜬 얼굴을 정의하고, 행번호 150

```
150 COLOR= 15: PLOT 15,13
```

은 행번호 170에서의 지연시간동안 왼쪽 눈을 감게 한다.

```
170 FOR DE = 1 TO 100: NEXT DE
```

윙크하는 시간은 FOR DE= 1 TO 100의 상한점 100을 그것보다 크게 하거나 작게 함으로써 느리게 또는 빨리하게 할 수 있다.

```
190 COLOR= 0: PLOT 15,13
```

은 행번호 210에 있는 FOR문에 의해 결정되는 시간 동안 화면의 배경색 (color = 0)으로 눈동자의 색을 바꿈으로서 눈을 다시 뜨게 한다.

```
210 FOR DE = 1 TO 100: NEXT DE
```

행 번호 300

```
300 GOTO 150
```

은 눈을 다시 감게 하기 위해서 행번호 150으로 보낸다.

이 프로그램은 GOTO문에 의해서 RUN이 실행되면 CTRL C 를 칠때까지 계속 수행되는데, 눈을 뜨고 감는 것을 백번만 수행하고 프로그램을 끝내게 해보면

```
135 FOR I = 1 TO 100
```

을 행번호 130과 140사이에 삽입하고 FOR-NEXT 루프이므로 행번호 300을

```
300 NEXT I
```

로 바꾼다.

저해상도 그래픽스에서의 색도표

저해상도 그래픽 모드에서의 색 코드가 실제로 어떤 색을 나타내고 있는가를 보기 위해 다음의 프로그램을 소개한다.

* PROGRAM 2.12

```
1 REM ************************
2 REM *  STANDARD            *
3 REM *    APPLE COLOR BARS  *
4 REM ************************
5 REM
10 GR : HOME
20 FOR I = 0 TO 31
30 COLOR= I / 2
40 VLIN 0,39 AT I
50 NEXT I
60 FOR I = 0 TO 14 STEP 2: PRINT
     TAB( I * 2 + 1);I;: NEXT I
70 PRINT
80 FOR I = 1 TO 15 STEP 2: PRINT
     TAB( I * 2 + 1);I;: NEXT I
90 PRINT : PRINT  TAB( 5);"STAND
     ARD APPLE COLOR BARS";
100 END
```

여기서 FOR…NEXT루프를 사용해서 색의 수 만큼 VLIN을 실행한다. 이때는 한 줄로 표시되는 것이 아니고 같은 색의 행번호 60과 80에 의해 두줄로 표시된다. 또 화면 하단의 TEXT부분은 2행으로 나누어져 색번호를 표시하고 그 밑에 「STANDARD APPLE COLOR BARS」란 프로그램 이름이 나타난다.

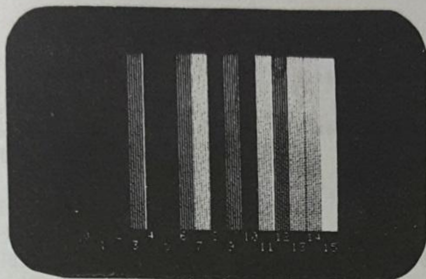


그림 2-18

확장된 GR 모드

지금까지 저해상도 그래픽 모드로 가로 40, 세로 40개로 전체 1600개의 블록을 나타낼 수 있었다. 이 때 맨 밑에 4줄의 텍스트를 사용할 수 있는데 만일 이 텍스트 부분이 없이 화면 전체를 그래픽으로 이용하고 싶다면 소프트 스위치를 전체 화면 모드로 하기 위해 다음 명령을 사용해야 한다.

```
* PROGRAM  2.13
  10  GR
  20  POKE  - 16302,0
```

이 두개의 명령으로 가로0~39, 세로 0~47 즉, 40×48으로 1920개의 블록을 나타낼 수 있다. 다음을 첨가시켜 보자.

```
30  FOR C = 0 TO 15
40  FOR X = 0 TO 19
50  COLOR= C
60  VLIN 0,47 AT 19 - X
70  VLIN 0,47 AT 20 + X
80  NEXT X
90  NEXT C
```

이 프로그램을 RUN시켜 보면 처음에는 텍스트의 잔재가 아래에 남아 있는 것을 알 수 있다. 이것은 GR 그래픽과 텍스트가 메모리의 같은 부분을 사용하여 화면에 표시되므로 이런 현상이 일어난다.

이렇게 확장된 저해상도 그래픽 모드를 수행시킨 후 다시 밑에 4줄의 텍스트를 쓰고 싶어졌을 때는 다음의 소프트 스위치인 POKE명령을 쓰면 된다.

```
100  POKE  - 16301,0
```

만일 그냥 GR 명령을 사용하면 지금 그래픽으로 나타난 화면이 전부 지워지므로 목적에 따라 GR을 사용할 것인지 POKE를 사용할 것인지를 결정해야 한다.

역시 그래픽에서 벗어나 일반 용도의 텍스트로 되돌아가려면 TEXT를 타이프하면 된다.

고해상도 그래픽스의 공이 튕기는 모양

고해상도 그래픽스로 공이 튕기는 모양을 시뮬레이션하는 프로그램에 대해 알아 보자.

* PROGRAM 2.14

```
1 REM **********************
2 REM *  BOUNCING  BALL    *
3 REM **********************
4 REM
10 DX = 4:X = 8:Y = 0:A = 2:V = 0

20  HOME : HGR
21  DATA   127,64,40,107,0,62,65,
     62
22  FOR I = 8207 TO 15645 STEP 10
     24
23  READ B: POKE I,B: NEXT I
24  DATA   70,73,73,73,70,64,80,7
     9
25  FOR I = 8209 TO 15647 STEP 10
     24
26  READ B: POKE I,B: NEXT I
27  DATA   41,41,111,41,47,0,1,63
     : FOR I = 8214 TO 15652 STEP
     1024: READ B: POKE I,B: NEXT
     I
28  DATA  127,1,127,8,127,62,65,6
     2: FOR I = 8216 TO 15654 STEP
     1024: READ B: POKE I,B: NEXT
     I
30  HCOLOR= 2: HPLOT 2,0 TO 2,133
     TO 266,133 TO 266,0
40  HCOLOR= 3: HPLOT X,Y: HPLOT X
     + 1,Y
50  IF Y = 132 THEN S =  PEEK ( -
     16336):V =  - V: IF V = 0 THEN
     V =  - 20
60  IF X > 263 THEN DX =  - DX
70  IF X < 6 THEN DX =  - DX
80 V = V + A
90  HCOLOR= 0: HPLOT X,Y: HPLOT X
     + 1,Y
100 Y = Y + V:X = X + DX
110  GOTO 40
```

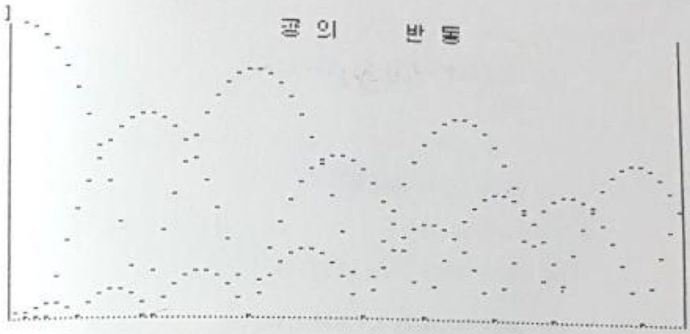
공은 행번호 40에서 그려지며, 보다 잘 보이게 하기 위해서 나란히 두개의 점을 이용하여 그려진다. 행번호 30에서는 두개의 벽과 하나의 바닥을 그려서 이곳에서 공이 튕기도록 한다. 공은 수평으로는 일정한 속도 DX에 따라 움직이고, 수직으로는 변수 V에 의해 가속도로서 움직인다.

그리고 공의 가속도는 일정한 값 A를 갖는다. 공이 움직이는 모양은 행번호 40에서 두개의 점을 흰색으로 그렸다가, 다음에 검은 색으로 행번호 90에서 다시 그림으로써 지워지도록 하여 시뮬레이트 되었다.

공의 위치 X, Y는 행번호 100에서 바뀌어, 그 위치에 새로운 공이 그려진다. 공이 벽이나 바닥에 튕기는 것은 행번호 50에서 70까지에서 제어된다. 그래픽을 디스플레이 할 때, 소리가 함께 나오면 보다 생동감있게 느껴질 것이다. 컴퓨터에는 스피커가 있으므로 행번호 50의 S=PEEK(−16336)에 의해 공이 반동하는 소리를 나타낼 수 있다. 같은 동작이 반복되도록 하기 위해서 행번호 50에서는 Y=132, V=0 로 공이 바닥에 정지하고 있으면, 다시 가속되어 동작을 계속하도록 하고 있다. 공은 CTRL-C 나 RESET 키에 의해 중지되기 전까지는 계속 반동동작을 반복하게 된다.

여기서 앞에서 배운 메모리 번지를 이용한 비트패턴으로 행번호 21에서 28까지의 프로그램에 의해서 화면 윗 부분에 「공의 반동」이라는 이름을 삽입시켰다.

이 프로그램에서 공이 움직이는 모양을 살펴보면 다음과 같다.



고해상도 그래픽을 이용한 SIN 함수 곡선

고해상도 그래픽 모드를 이용하여 SIN 그래프를 그릴 수 있다. 함수는 독립변수 X에 대해서 종속변수 Y가 항상 하나의 값을 가지므로 X를 적당히 변화시키면서 Y값은 시스템내에 내장되어 있는 SIN(X) 함수를 사용하여 주면 된다.

여기서 가장 신경을 써야 할 문제는 좌표를 이동시키는 일이다.

우선 함수의 좌표축을 Y축은 X=15으로, X축은 Y=70인 줄로 잡아서, 이 가로, 세로축이 SIN 함수를 그리는 기본축이 된다. 즉 이축을 기본으로 하여 SIN 그래프를 옮겨 그리는 것이다.

```
100 HGR
110 HCOLOR= 3
120 HPLOT 0,70 TO 279,70
130 HPLOT 15,0 TO 15,145
```

만일 당신의 TV 방식 때문에 세로줄이 나타나지 않으면 한 줄을 더 첨가하면 해결된다.

```
131 HPLOT 16,0 TO 16,145
```

이제 SIN 함수의 최대, 최소치를 화면에 어떻게 나타내는가 하는 것이 문제가 된다. 원래 SIN 함수의 최대치는 1이고 최소치는 -1이지만 여기서는 좌표이동과 확대가 일어나고 있으므로 HGR에서의 40개의 점을 1에 해당하도록 하면, 가로축이 Y=70이므로 이보다 40개의 점 위의 좌표는 Y=30으로 최대치가 되고, 아래로 Y=110이 최소치가 된다. 보기 쉽게 하기 위해서는 최대치와 최소치를 가로 점선으로 찍어 보자.

```
140 FOR JUMSUM = 0 TO 270 STEP 1
    0
150 HPLOT JUMSUM,30
160 HPLOT JUMSUM,110
180 NEXT JUMSUM
```

이 때 점선이 좀 부드럽게 혹은 여유있게 찍히게 하려면 루프내에 지연시키는

부분을 다음과 같이 삽입하면 된다.

```
170  FOR PAUSE = 1 TO 50: NEXT PA
     USE
```

여기서 그래프를 그리기 전에 잠시 쉬게 하는 방법으로

```
200  FOR P = 1 TO 300: NEXT P
```

그 다음 INPUT문을 사용하여 어느 색으로 그릴 것인가를 묻게 하자.

```
400  PRINT "WHICH COLOR DO YOU WA
     NT?"
410  INPUT "(1.GREEN 2.VIOLET 3.W
     HITE 4.ORANGE 5.BLUE)";C
420  HCOLOR= C
```

이제 진폭을 조절하는 기능을 첨가하자.

```
430  INPUT "AMPLITUDE? (0-1.75)";
     A
480 A = A * 40
```

즉 진폭을 점선부분으로 1로 잡았을 때 그 비율을 0～1.75배로 표시해 준다. 그런데 여기서 1은 HGR의 화면상에서 40에 해당되므로 40을 곱해 준다. 그러면 1.75배의 진폭은 자연히 진폭 70으로써 화면에 표시된다.

2π와 π를 가로축에 표시하기 위해 다음을 첨가하자.

```
500  HPLOT 140,67 TO 140,73: HPLOT
     265,66 TO 265,74
```

실제로 SIN 그래프를 찍는 부분을 만들자.

```
510  FOR N = 1 TO 250
520 X = (2 * 3.14159 / T) * N
530 Y =  - A *  SIN (X) + 70
540  HPLOT N + 15,Y
550  NEXT N
```

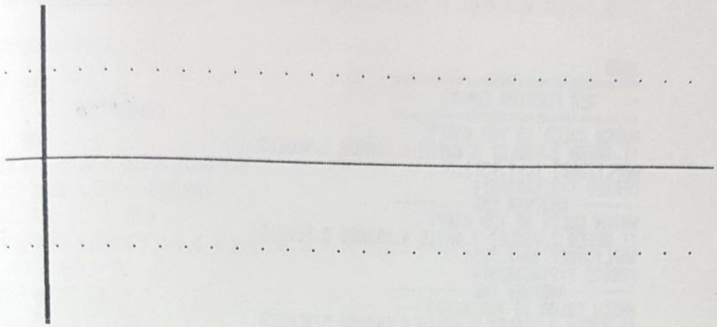
여기서 HGR의 40점을 1의 비율로 배정했기 때문에 주기 T는 거기다 40을 곱해 주어야 화면의 비율과 같게 된다.

```
490  INPUT "PERIOD T=?(RADIAN)";T

495 T = T * 40
```

입력되는 모든 각은 라디언이므로 만일 주기 360°의 SIN파를 원하면 2π인 6.28을 입력시켜야 한다.

그럼 이 프로그램을 실행시켜 보자. 화면에는 다음과 같은 기본 좌표축이 나타나며, 색과 진폭등을 묻게 될 것이다.



한가지 SIN파 위에 주기와 진폭과 색을 바꾸어가며 계속 그려보고 싶으면 다음을 첨가한다.

```
600  PRINT "--------ANOTHER ONE -
     -------"
610  GOTO 400
```

지금까지의 프로그램을 종합하여 완성된 프로그램을 만들면 다음과 같다. 이 프로그램의 제목을 SIN Fuction curve라고 하자.

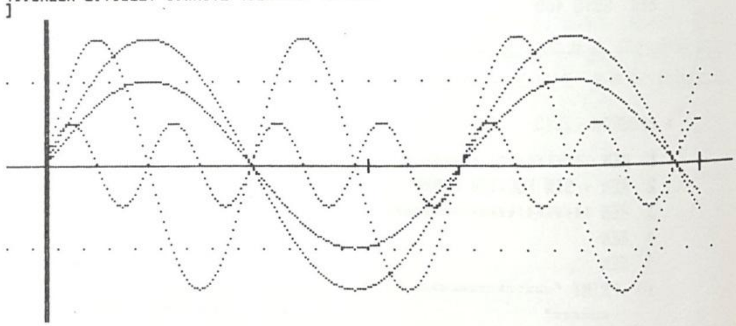
* PROGRAM 2.15

```
1  REM **********************
2  REM * SIN FUCTION CURVE  *
3  REM **********************
4  REM
5  REM
10  PRINT "=====================
    ======"
20  PRINT "==   SIN FUCTION CURVE
      ==="
30  PRINT "=====================
    ======"
100  HGR
110  HCOLOR= 3
120  HPLOT 0,70 TO 279,70
130  HPLOT 15,0 TO 15,145
131  HPLOT 16,0 TO 16,145
140  FOR JUMSUM = 0 TO 270 STEP 1
     0
150  HPLOT JUMSUM,30
160  HPLOT JUMSUM,110
170  FOR PAUSE = 1 TO 50: NEXT PA
     USE
180  NEXT JUMSUM
200  FOR P = 1 TO 300: NEXT P
400  PRINT "WHICH COLOR DO YOU WA
     NT?"
410  INPUT "(1.GREEN 2.VIOLET 3.W
     HITE 4.ORANGE 5.BLUE)";C
420  HCOLOR= C
430  INPUT "AMPLITUDE? (0-1.75)";
     A
480 A = A * 40
490  INPUT "PERIOD T=?(RADIAN)";T

495 T = T * 40
500  HPLOT 140,67 TO 140,73: HPLOT
     265,66 TO 265,74
510  FOR N = 1 TO 250
520 X = (2 * 3.14159 / T) * N
530 Y =  - A *  SIN (X) + 70
540  HPLOT N + 15,Y
550  NEXT N
600  PRINT "--------ANOTHER ONE -
     -------"
610  GOTO 400
```

자, 이 프로그램이 완성되었으면 RUN을 타이프하고 **RETURN** 키를 누르자.

```
]RUN
=============================
==   SIN FUCTION CURVE   ===
=============================
WHICH COLOR DO YOU WANT?
(1.GREEN 2.VIOLET 3.WHITE 4.ORANGE 5.BLUE)3
AMPLITUDE? (0-1.75)1.5
PERIOD T=?(RADIAN)4
--------ANOTHER ONE --------
WHICH COLOR DO YOU WANT?
(1.GREEN 2.VIOLET 3.WHITE 4.ORANGE 5.BLUE)3
AMPLITUDE? (0-1.75)1
PERIOD T=?(RADIAN)4
--------ANOTHER ONE --------
WHICH COLOR DO YOU WANT?
(1.GREEN 2.VIOLET 3.WHITE 4.ORANGE 5.BLUE)3
AMPLITUDE? (0-1.75)1.5
PERIOD T=?(RADIAN)2
--------ANOTHER ONE --------
WHICH COLOR DO YOU WANT?
(1.GREEN 2.VIOLET 3.WHITE 4.ORANGE 5.BLUE)3
AMPLITUDE? (0-1.75)0.5
PERIOD T=?(RADIAN)1
--------ANOTHER ONE --------
WHICH COLOR DO YOU WANT?
(1.GREEN 2.VIOLET 3.WHITE 4.ORANGE 5.BLUE)
]
```



이 프로그램을 이용한다면 그 주기와 진폭의 개념을 직접 눈으로 확인할 수 있을 것이다. 당신은 이 기본 프로그램을 응용하여 자유자재로 바꾸어 그릴 수 있을 것이다.

사각형 화상 변환

다음의 프로그램은 DATA문을 이용하여 8개의 사각형을 그리는 것이다. 좌표 (X, Y)를 읽어 화면 좌표(SX, SY)로 변형시켜 그것에 따라 점이나 선을 긋고 있다. 각 사각형의 크기, 모양, 위치는 변수 A, B, C, D의 값을 바꿈으로써 변형될 수 있다.

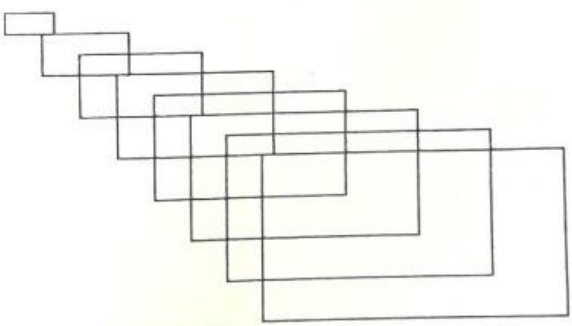
여기서 두가지 형태의 변형이 사용되었다. 그 하나는 각 좌표값들은 변수 A와 C로 곱해져 그 크기가 변형된다. 이것을 화상에 대한 **크기조절**(scaling)이라 한다. 또 하나는 화상의 위치는 변수 B와 D의 값에 의해 변경된다. 이것을 **선형이**

동(translating)이라고 한다.

```
* PROGRAM  2.16

1  REM
        RECTANGLES
2  REM
10 A = 20:B = 0:C = 10:D = 0
20  HOME : HGR : HCOLOR= 3
30  FOR J = 1 TO 8
40  READ X,Y:SX = A * X + B:SY =
     C * Y + D
50  HPLOT SX,SY
60  FOR I = 1 TO 4
70  READ X,Y:SX = A * X + B:SY =
     C * Y + D
80  HPLOT  TO SX,SY
90  NEXT I
100 A = A + 15:B = B + 15:C = C +
     10:D = D + 10
110  RESTORE
120  NEXT J
130  DATA  0,0,1,0,1,1,0,1,0,0
140  END

]
```



HGR 2에서의 그래프의 SHIFTING과 SCALING

손으로 그린 그래프는 보통 가로로 그린 X축과 세로로 그린 Y축과 많은 점을 연결하여 그린 연속적인 곡선으로 이루어져 있다. 컴퓨터로 그래프를 그리는 데에는 몇가지 명백한 제한이 있다.

즉 화면의 그래프 구역을 그리는데 쓸 점들의 수가 제한되어 있다. 그래서 쓸 수 있는 그래프 점들의 수를 초과하게 값을 내는 함수를 조절하기 위해 X축과 Y축을 적당히 조정할 필요가 있다.

이 문제는 함수 Y=f(x)에서 X의 최대치와 최소치 영역과 Y의 최대치와 최소치의 범위를 결정한 다음, 양축을 적당히 축소 확대함으로써 해결된다. 이러한작업을 Shifting 과 Scaling이라 부른다.

시프팅(Shifting)은 각 Y값에서 Y의 최소치를 빼서 이루어진다. 예로써 어떤 함수가 −10과 15사이의 값을 가질때 Y−(−10)는 그래프를 Y축을 따라 0과 25사이의 점들로 이동시킨다. 이는 그래프의 모양에는 영향을 주지 않는다.

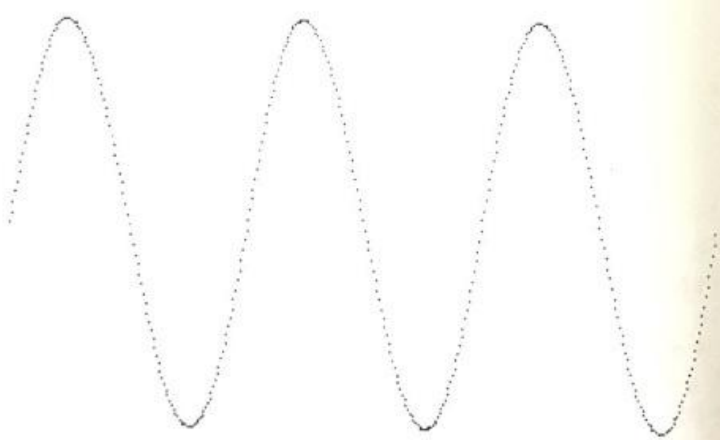
한편 스케링(Scaling)은 최대치와 최소치의 차이값을 취한 다음, 이 차이값을 지정한 축에 있을수 있는 점들의 수보다 하나 작은 수로 나눈다. 하나의 여분의 공백은 그 축이 가장 작은 값에서 시작할 수 있도록 해준다. 예로써 만일 어느 그래프의 Y값의 최대치와 최대치 사이의 간격이 400단위이고 화면을 그리는데 가로로 192개의 점밖에 없다면 400을 (192 − 1)혹은 191로 나누어 적당한 규격을 만든다. 대략으로 그래프의 Y축 위 매 2점마다 화면의 1점과 대응하게 된다.

다음 프로그램은 함수 Y=150 ∗SIN(X)가 고해상도 화면에 맞게 어떻게 시프트되고 스케일 되는가를 보여주고 있다.

```
* PROGRAM  2.17

10  HGR2 : HCOLOR= 3:PI = 3.14159

20  FOR X =  - 3 * PI TO 3 * PI STEP
     .05
30 Y = 150 *  SIN (X)
40 Y0 = Y / (300 / 191)
50 Y0 = Y0 + 96
60 X0 = X / (6 * PI / 279)
70 X0 = X0 + 140
80  HPLOT X0,Y0
90  NEXT X
```



함수는 행번호30에서 행번호20에 의해 증가치가 0.05 로 -3π에서 3π까지로 계산된다.

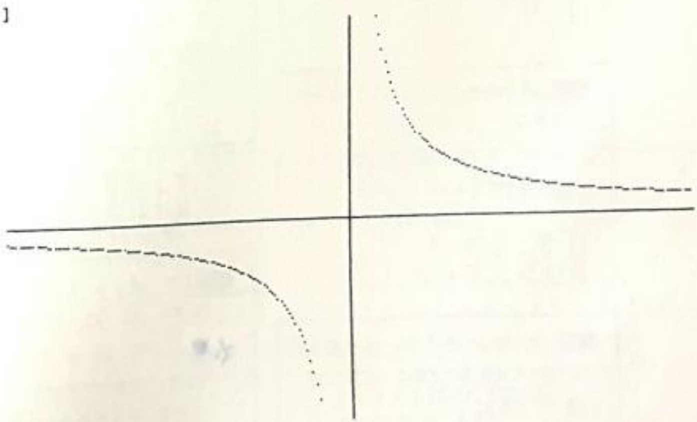
결과 Y0는 Y를 300/191비율로 나눔으로 스케일된다. 300은 이 함수의 절대 범위를 가리킨다. 191은 화면의 세로폭을 말한다. 왜냐하면 세로폭은 0에서 부터 시작하기 때문이다. X는 행번호60에서 마찬가지로 스케일된다.

X와 Y는 둘다 행번호70, 50에 의해 각각 시프트된다.

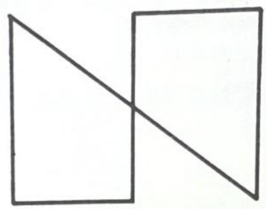
다음 프로그램은 화면에 함수 Y=1/X을 그린다. 어떻게 시프트되고 스케일되는가를 주시하기 바란다.

```
* PROGRAM  2.18
 10  HGR2 : HCOLOR= 3
 20  HPLOT 140,0 TO 140,191
 30  HPLOT 0,96 TO 279,96
 40  FOR X =  - 12 TO 12 STEP .1
 50  IF X = 0 THEN 140
 60 Y = 1 / X
 70 X1 = X / (24 / 279)
 80 X1 = X1 + 140
 90 Y1 = Y / (2 / 191)
 100 Y1 = 96 - Y1
 110  IF X1 < 0 OR X1 > 279 THEN 1
      40
 120  IF Y1 < 0 OR Y1 > 191 THEN 1
      40
 130  HPLOT X1,Y1
 140  NEXT X

 ]
```

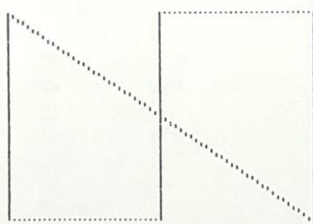


문제 115. 고해상도 그래픽을 사용하여 다음 그림을 그리는 프로그램을 작성하라.



```
10  HGR : HCOLOR= 6
20  HPLOT 10,10 TO 130,110 TO 130
    ,10
30  HPLOT  TO 70,10 TO 70,110
40  HPLOT  TO 10,110 TO 10,10

]
]
```



문제 116. 화면좌표 점(15, 18) 에 윗부분을 둔 약 1인치 길이의 자리에 소문자 i를 그리는 프로그램을 작성하라.

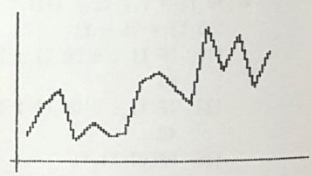
```
10  GR : COLOR= 14
20  PLOT 15,18
30  VLIN 20,25 AT 15
```

문제 117. 다음 데이타를 이용한 꺾은선 그래프를 그리는 프로그램을 작성하라.

DATA 125, 160, 180, 120, 140, 125, 130, 190, 200, 180, 160, 250, 200, 240, 180, 220

```
10  HCOLOR= 3
20  HGR
30  HPLOT 18,80 TO 130,80
40  HPLOT 20,10 TO 20,85
50  DIM A(16)
60  FOR J = 1 TO 16
70  READ A(J)
80  NEXT J
90  FOR I = 1 TO 15
100 Y = 118 - A(I) * .4
110 Y1 = 118 - A(I + 1) * .4
120  HPLOT (I - 1) * 6 + 24,Y TO
     I * 6 + 24,Y1
130  NEXT I
140  END
150  DATA  125,160,180,120,140,12
     5,130,190,200
160  DATA  180,160,250,200,240,18
     0,220

]
]
```

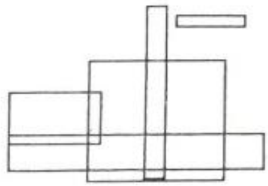


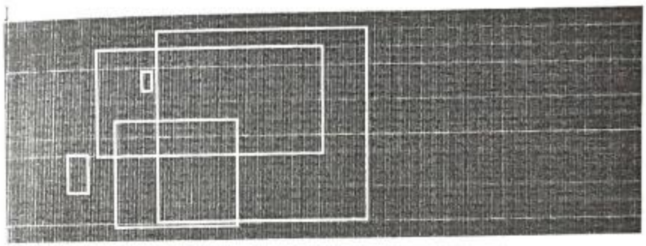
문제 118. 오렌지색으로 된 사각형을 (19, 10)의 위치에 길이가 3개의 열, 높이가 2개의 행이되도록 그리는 프로그램을 작성하라.

```
10  GR : COLOR= 9
20  HLIN 19,21 AT 10
30  HLIN 19,21 AT 11
```

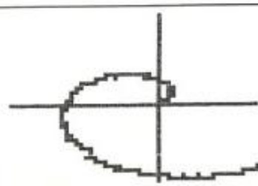
문제 119. 난수를 4개 만들고 각각 X1, X2, Y1, Y2로 하여 (X1, X2), (X2, Y2)을 대각선으로하는 사각형을 5개 만드는 프로그램을 작성하라. 단 크기에는 제한이 없다.

```
10  HCOLOR= 3
20  HGR
30  FOR I = 1 TO 5
40 X1 =  RND (1) * 1000
50  IF X1 > 159 THEN 40
60 Y1 =  RND (1) * 1000
70  IF Y1 > 99 THEN 60
80 X2 =  RND (1) * 1000
90  IF X2 > 159 THEN 80
100 Y2 =  RND (1) * 1000
110  IF Y2 > 99 THEN 100
120  HPLOT X1,Y1 TO X2,Y1 TO X2,Y
    2 TO X1,Y2 TO X1,Y1
130  NEXT I
140  END
```

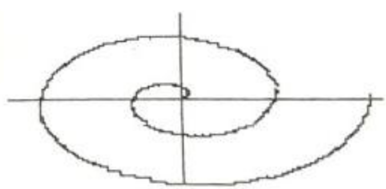




문제 120. 아르키메데스의 회돌이(渦卷)를 만드는 프로그램을 작성하라. 아르키메데스 회돌이란 오른쪽 그림과 같은 동경 $R=A\theta$(A는 정수)인 점의 궤적이다.



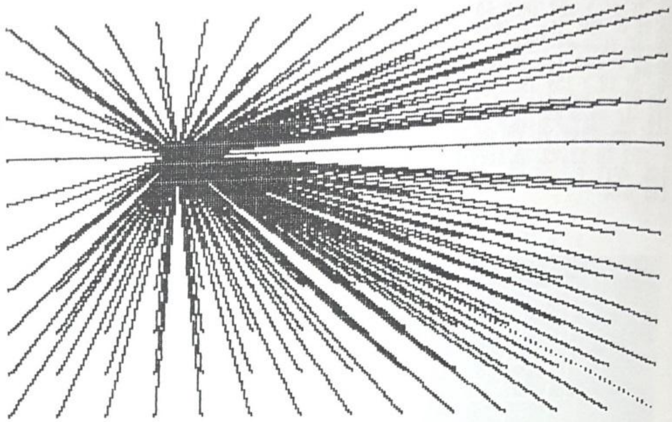
```
10  HCOLOR= 3
20  HGR
30  FOR X = 0 TO 4 * 3.14159 STEP
    .1
40 R = 6 * X
50 X1 =  COS (X) * R + 80
60 Y1 = 60 -  SIN (X) * R * .6
70 X2 =  COS (X + .1) * R + 80
80 Y2 = 60 -  SIN (X + .1) * R *
    .6
90  HPLOT X1,Y1 TO X2,Y2
100  NEXT X
110  HPLOT 10,60 TO 159,60
120  HPLOT 80,20 TO 80,99
130  END
```



문제 121. 좌표(70, 60)을 중점으로 퍼져나가는 선을 HGR2를 이용하여 프로그램을 작성하라. 단 컬러도 HCOLOR=1에서 7까지 변하도록 할 것.

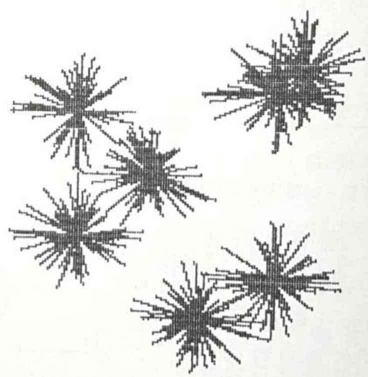
```
10  HGR2
20  FOR C = 1 TO 7
30  HCOLOR= C
40  FOR X = 1 TO 279 STEP 20
50  FOR Y = 1 TO 191 STEP 20
60  HPLOT 70,60 TO X,Y
70  NEXT Y,X,C
80  GOTO 10
```

]



문제 122. 다음의 그래픽을 HGR 2 모드로 나타내는 프로그램을 작성하라.

]

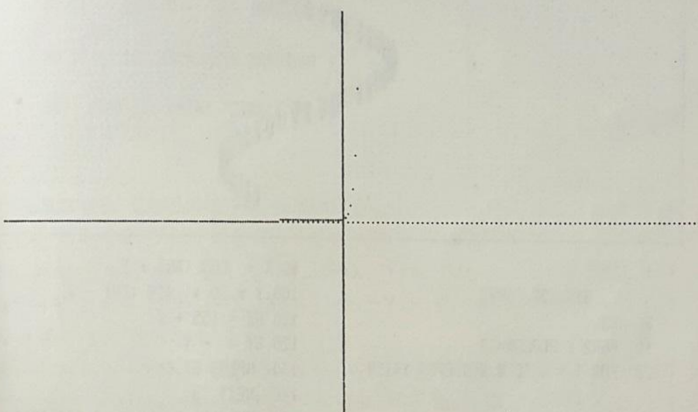


```
1  REM
       FIRE FLOWER
2  REM
10  HOME : HGR2 : HCOLOR= 7
20  FOR I = 1 TO 7
30  READ CX,CY
40  FOR TH = 0 TO 6 STEP .07
50 X = 30 *  RND (1) *  COS (TH)
60 Y = 30 *  RND (1) *  SIN (TH)
70  HPLOT CX,CY TO CX + X,CY + Y
80  NEXT TH
90  NEXT I
100  DATA  95,55,95,115,125,85,17
     5,135
110  DATA  145,155,185,35,175,45
```

문제 123. $Y=2^x$의 그래프를 HGR2 로 그리는 프로그램을 작성하라.

```
10  HGR2 : HCOLOR= 5
20 X0 = 135:Y0 = 100
30  HPLOT 0,Y0 TO 279,Y0
40  HPLOT X0,0 TO X0,191
50  HCOLOR= 3
60  FOR X1 = 0 TO 279
70 X = X1 - X0
80 Y = 2 ^ X
90 Y1 = Y0 - Y
100  IF Y1 > = 0 AND Y1 < = 191
      THEN  HPLOT X1,Y1
110  NEXT X1

]
```

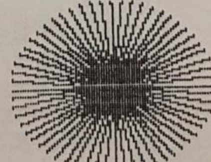


문제 124. 시계의 분침과 같이 1분동안 1회전하는 프로그램을 작성하라.

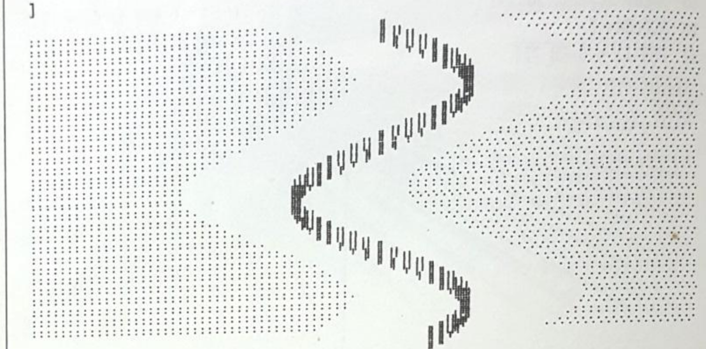
```
10  HCOLOR= 3
20  HGR
30  DEF  FN A(X) =  COS (X / 180 *
     3.14159) * 40 + 79
40  DEF  FN B(X) = 49 -  SIN (X /
     180 * 3.14159) * 40
50  FOR X = 84 TO  - 270 STEP  -
     6
60  FOR Z = 0 TO 430
70  NEXT Z
80  HPLOT 79,49 TO  FN A(X), FN B
     (X)
90  NEXT X
100  END
```

#60～#70
Timer 설정

```
]
```



문제125. 다음의 도형은 SIN 및 COS 곡선을 변화시킨 그래픽으로 곡선은 HCOLOR= 3으로 나타내고 주위는 점선으로 둘러싸게 하는 프로그램을 작성하라.



```
1  REM
     SIN,COS CURVE
2  REM
10  HGR2 : HCOLOR= 3
20  FOR Z = 0 TO 8.8 STEP 3.14159
     / 20
30 I = 34 *  SIN (Z)
40 J = Z * 50 / 3.14159
50  FOR K = 0 TO 110 + I STEP 3
60  HPLOT K,10 + J
70  NEXT K
80  FOR TH = 3.64 TO 5.8 STEP 3.1
     4159 / 90
90 X =  COS (TH) + I
100 Y = 20 *  SIN (TH) - J
110 GX = 155 + X
120 GY =  - Y
130  HPLOT GX,GY
140  NEXT TH
150  FOR L = 200 + I TO 279 STEP
     3
160  HPLOT L,10 + J
170  NEXT L
180  NEXT Z
190  END
```

2-4 곡선 그래픽의 응용

SIN, COS 함수를 이용한 원

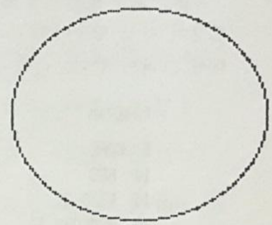
애플을 사용하여 원을 그리는 여러가지 방법에 대해 알아보자.

가장 쉽게 떠오르는 생각은 SIN과 COS함수를 이용해서 좌표를 구하고 그 점을 계속 표시하는 방법이다.

* PROGRAM 2.19

```
5  HOME
10  HGR
15  VTAB 21
20  HCOLOR= 3
30 A = 140
40 B = 80
50 R = 50
60  FOR ALPHA = 0 TO 360
70 RADIAN = 3.141592 * ALPHA / 18
     0
80 X1 =  INT (R *  COS (RADIAN) +
     .5)
90 Y1 =  INT (R *  SIN (RADIAN) +
     .5)
100  HTAB 13: PRINT "CIRCLE BY SI
     N & COS";
110  HPLOT A + X1,B + Y1
120  NEXT ALPHA
130  PRINT : PRINT

]
```



대칭이용법에 의한 원

이 프로그램을 실행시키면 시간이 많이 걸린다. 그러면 다른 성질을 이용해 보자. 원은 좌우상하가 서로 대칭이다. 원의 한점을 알고 있다면 X축에 대칭인 점, Y축에 대칭인 점이 각각 원의 점이다. 즉(X, Y)가 원의 한 점이라면(X, −Y)도 마찬가지일 것이다. 또한 (−X, Y)와 (−X,−Y)도 원의 한 점이다. 이렇게 하면 일이 4분의 1로 줄어든다.

따라서 원은 Y=X라는 직선과도 대칭이다. 그러므로(X, Y)가 원의 한 점이라면 (Y, X)도 원의 한 점이다. 똑같은 이유로 해서 (Y, −X), (−Y, X), (−Y, −X)도 원의 한 점이다.

이제 원의 한 점을 알면 원의 여덟개의 점을 알게 된 것이 된다. 이것을 이용한 것이 다음의 프로그램이다.

* PROGRAM 2.20

```
5  HOME
10  HGR
15  VTAB 21
20  HCOLOR= 3
30 A = 140
40 B = 80
50 R = 50
60  FOR ALPHA = 0 TO 45
70 RADIAN = 3.141592 * ALPHA / 18
     0
80 X1 =  INT (R *  COS (RADIAN) +
     .5)
90 Y1 =  INT (R *  SIN (RADIAN) +
     .5)
100 XY = PY - X1 - X1 + 1
105  HTAB 7: PRINT "CIRCLE BY SYM
     METRICAL FIGURE";
110  HPLOT A + X1,B + Y1
120  HPLOT A - X1,B + Y1
130  HPLOT A + X1,B - Y1
140  HPLOT A - X1,B - Y1
150  HPLOT A + Y1,B + X1
160  HPLOT A - Y1,B + X1
170  HPLOT A + Y1,B - X1
180  HPLOT A - Y1,B - X1
190  NEXT ALPHA
200  PRINT : PRINT
```

하지만 대칭 이용법에 의한 원을 그리는 방법은 아직도 시간이 꽤 걸린다.

원의 방정식을 이용한 원

이제 SIN과 COS 대신에 원의 방정식인

$X^2+Y^2=R^2$

을 이용해 보자. 이대로는 불편하므로,

Y=SQRT(R*R-X*X)

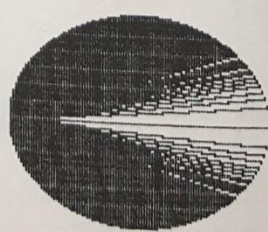
를 이용한다.

X를 -R부터 R까지 변화시키면서 Y를 찾을 수 있다. 하지만 이 식으로는 루트안이 항상 양수여야 하므로 양수값 Y밖에 구할 수 없다. 따라서 -Y값도 고려해서 그려야 한다. 다음의 프로그램이 원의 방정식을 이용한 것이다.

```
* PROGRAM  2.21

 5  HOME
 10  HGR
 15  VTAB 21
 20  HCOLOR= 3
 30 A = 140
 40 B = 80
 50 R = 50
 60 X0 =  - R:Y0 = 0
 70  FOR X1 =  - R TO R
 80 Y1 =  INT (.5 +  SQR (R * R -
      X1 * X1))
 100  HTAB 12: PRINT "CIRCLE BY AN
       EQUATION";
 110  HPLOT A + X0,B + Y0 TO A + X
      1,B + Y1
 120  HPLOT A + X0,B - Y0 TO A + X
      1,B - Y1
 130  NEXT X1
 140  PRINT : PRINT

]
```



원의 방정식 대칭법을 혼합한 원

이것은 SIN과 COS이용법보다 빠르지만 여전히 느리다. 그러면 방정식과 대칭법을 혼합하여 이용하면 좀 더 빠른 방법을 얻을 수 있다. 다음은 원의 방정식과 대칭 이용법에 의한 프로그램이다.

* PROGRAM 2.22

```
5  HOME
10  HGR
15  VTAB 21
20  HCOLOR= 3
30 A = 140
40 B = 80
50 R = 50
60 X0 =  - R:Y0 = 0
70  FOR X1 =  - R TO R
80 Y1 =  INT (.5 +  SQR (R * R -
     X1 * X1))
100  HTAB 7: PRINT "CIRCLE BY SYM
     M.FIGURE & EQUATION";
110  HPLOT A + X0,B + Y0 TO A + X
     1,B + Y1
120  HPLOT A - X0,B + Y0 TO A - X
     1,B + Y1
130  HPLOT A + X0,B - Y0 TO A + X
     1,B - Y1
140  HPLOT A - X0,B - Y0 TO A - X
     1,B - Y1
150  HPLOT A + Y0,B + X0 TO A + Y
     1,B + X0
160  HPLOT A - Y0,B + X0 TO A - Y
     1,B + X0
170  HPLOT A + Y0,B - X0 TO A + Y
     1,B - X0
180  HPLOT A - Y0,B - X0 TO A - Y
     1,B - X0
190 X0 = X1:Y0 = Y1
200  IF  - X1 >  = Y1 THEN X1 = X
     1 + 1: GOTO 80
210  PRINT : PRINT
```

포텐셜함수에 의한 원

이 방법으로는 반경이 작은 원은 빠르게 그리지만 큰 원은 웬만큼 시간이 걸린다. 이보다 더 빠르게 그리는 방법은 없을까?

화면은 수많은 점으로 이루어져 있다. 각 점은 좌표(X, Y)로 표시가 가능하다. 하지만 X, Y는 항상 정수이다.

```
]HPLOT 0.5,0.5
```

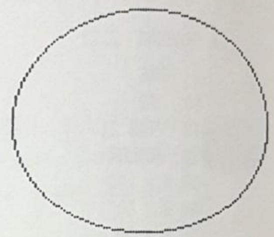
로 해도 점이 찍히지만 이는 반올림을 해서 정수로 만든 것이다. 당신이 화면의 원위 한 점에 있다고 가정할 때, 당신의 다음 점을 그리기 위해 나갈수 있는 방향은 8개밖에 없다.

원을 4등분해 보자. 4등분된 원의 한 점에서 보면 나갈수 있는 방향은 단지 3가지 뿐이다. 만약 8등분을 해 본다면 방향은 두가지 뿐이다. 예를들어 당신이 (R, 0)의 점에 있다면, 다음 점은 (R－1, 0) 또는 (R－1, 1)이다.

* PROGRAM 2.23

```
5  HOME
10  HGR
15  VTAB 21
20  HCOLOR= 3
30 A = 140
40 B = 80
50 R = 50
60 PHI = 0
70 Y1 = 0
80 X1 = R
90 PY = PHI + Y1 + Y1 + 1
100 XY = PY - X1 - X1 + 1
105  HTAB 7: PRINT "CIRCLE BY POT
     ENTIAL FUNCTION";
110  HPLOT A + X1,B + Y1
120  HPLOT A - X1,B + Y1
```

```
130  HPLOT A + X1,B - Y1                 ]
140  HPLOT A - X1,B - Y1
150  HPLOT A + Y1,B + X1
160  HPLOT A - Y1,B + X1
170  HPLOT A + Y1,B - X1
180  HPLOT A - Y1,B - X1
190 PHI = PY
200 Y1 = Y1 + 1
210  IF  ABS (XY) <  ABS (PY) THEN
     PHI = XY:X1 = X1 - 1
220  IF X1 > = Y1 THEN 90
230  PRINT : PRINT
```



이러한 방법을 포텐셜함수에 의한 원의 작도라 할 수 있다. 화면에는 정수 좌표의 점만을 그리므로 항상 조그만큼의 에러는 있다. 이것은 지금까지 설명한 것 중에는 가장 빠르게 원을 그린다.

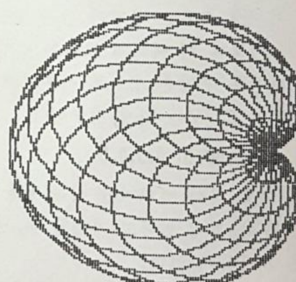
심장모양의 곡선

그러면 원을 이용하여 다른 형태의 곡선을 그려보자.

* PROGRAM 2.24

```
1  REM ********************** ]
2  REM *  C A R D I O I D   *
3  REM **********************
4  REM
10 PI = 3.14159266
20 X0 = 25:Y0 = 0:XN = X0:YN = Y0

30 C =  COS (.1):S =  SIN (.1):N =
   64
40 C1 =  COS (PI / 10):S1 =  SIN
   (PI / 10)
50  HGR2 : HCOLOR= 3
60  FOR I = 1 TO 19
70 T = XN * C1 - YN * S1
80 YN = XN * S1 + YN * C1:XN = T
90 CX = 140 + XN:CY = 96 - YN
100 X = X0 - XN:Y = Y0 - YN
110  GOSUB 140
120  NEXT I
130  END
140  FOR J = 1 TO N
150 T = X * C - Y * S
160 Y = Y * C + X * S:X = T
170 SX = X + CX:SY = CY - Y
180  IF FL = 0 THEN  HPLOT SX,SY
190  HPLOT  TO SX,SY:FL = 1
200  NEXT J
210  RETURN
```



이 프로그램에서는 19개의 원이 그려지는데, 각 원의 중심은 기준원상(base circle)에 위치한다. 그리고 각 원들은 다같이 기준원상의 공통점P를 지난다. 기준원의 원주는 20개의 점들 (P점과 19개의 중심점)에 의해 똑같은 간격으로 나뉘어지는데, 이 기준원은 그려지지 않는다. 따라서 이 프로그램에 의해 나타나는 그래픽은 마치 심장모양의 곡선이 된다.

행번호 60에서 120까지는 원주상의 점을 **순환방법**(recursive method)을 이용하여 구해 19개의 중심점(CX, CY)들을 정의한다. 각 중심점에 대하여 행번호 140~210은 공통점 P를 지나는 원을 다시 순환 방법을 이용하여 그린다.

순환식을 이용한 반복법에 의한 원

여기서 **순환식** (recursive generation)을 이용한 반복법에 의한 원의 작도를 생각해 보자. SIN과 COS함수를 이용하는 방법에서 매번 COS과 SIN값을 구하기 위해 많은 시간이 소요된다. 그러므로 여기에서 다루어질 방법에는 단지 한 각도에 대해서만 SIN값과 COS값을 구하면 된다.

먼저 원을 그리는데 사용되는 점들에 번호를 붙여 생각해 보기로 하자. 이 점들에 다음과 같이 순서대로 번호를 줄 수 있을 것이다.

(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), …, (X_n, Y_n)

두개의 연속된 점(X_n, Y_n)와 (X_{n+1}, Y_{n+1})사이에는 다음과 같은 관계가 성립된다.

$X_n = R\ COS((\theta)$

$Y_n = R\ SIN(\theta)$

$X_{n+1} = R\ COS(\theta + D\theta)$

$Y_{n+1} = R\ SIN(\theta + D\theta)$

여기서 $D\theta$ 는 점을 구하는 루프에서의 증가분을 나타낸다. 이것들을 삼각함수를 이용하여 다시 정리하면 다음과 같은 관계식을 얻을 수 있다.

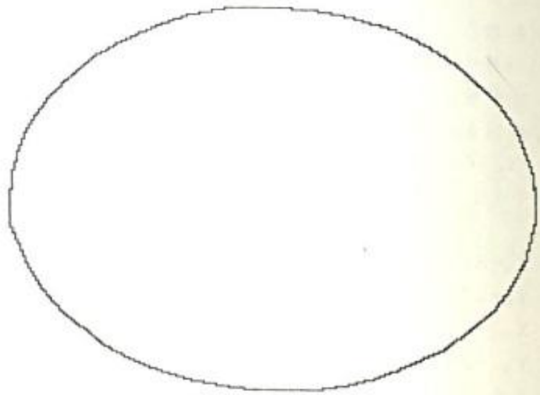
$$X_{n+1} = R\ COS(\theta + D\theta) = R\ COS(\theta)COS(D\theta) - R\ SIN(\theta)SIN(D\theta)$$
$$= X_n COS(D\theta) - Y_n SIN(D\theta)$$
$$Y_{n+1} = R\ SIN(\theta + D\theta) = R\ SIN(\theta)\ COS(D\theta) + R\ COS(\theta)\ SIN(D\theta)$$
$$= Y_n\ COS(D\theta) + X_n\ SIN(D\theta)$$

그러므로 이 관계식으로 부터 단지 일정한 값 $COS(D\theta)$와 $SIN(D\theta)$를 한번만 계산하여 구해놓고, 초기값 X_n과 Y_n이 주어지면, 다음에 그려질 점 (X_{n+1}, Y_{n+1})을 구할 수 있다. 이와같은 과정을 다음의 프로그램에 나타내었다. 프로그램에서는 $D\theta$대신에 DT, $COS(D\theta)$ 대신 C, $SIN(D\theta)$ 대신 S를 각각 사용하였다. 여기서 위식은 행번호 50에서 T=X*C−Y*S : Y=Y*C+X*S : X=T로 표현한다.

```
* PROGRAM  2.25
 10 CX = 140:CY = 96:SC = 1.16:X =
      90:Y = 0:FL = 0
 20 DT = .1:C =  COS (DT):S =  SIN
      (DT):N = 6.4 / DT
 30  HGR2 : HCOLOR= 3
 40  FOR I = 1 TO N
 50 T = X * C - Y * S:Y = Y * C +
      X * S:X = T
 60 SX = SC * X + CX:SY = CY - Y
 70  IF FL = 1 THEN 90
```

```
80  HPLOT SX,SY:FL = 1
90  HPLOT  TO SX,SY
100  NEXT I

]
```



이제까지의 원들은 모두 중심이 원점(0,0)에 있는 것들인데, 중심이 어떤 점 (H, K)에 있는 원을 구하려면 계산에 의해 구해지는 각 점들을 선형 이동시켜 주면 된다. 각 점(X, Y)는 다음의 행렬곱에 의하여 (X+H, Y+H)로 이동된다.

$$(X,\ Y,\ 1)\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ H & K & 1\end{pmatrix} = (H+X,\ Y+K,\ 1)$$

위의 프로그램에서는 단지 행번호 60을 다음과 같이 수정함으로써 선형 이동효과를 얻을 수 있다.

```
60 SX = SC * (T + H) + CX:SY = CY
     - (Y + K)
```

원으로부터 변형된 타원

타원은 원이 변형된 것으로 생각할 수 있는데, 원을 X방향이나 Y방향으로 확대 또는 축소시킴으로써 타원을 작도할 수 있다.

예를 들어, 반지름이 1인 원, $X^2+Y^2=1$의 X좌표의 값은 A배로 크기를 조절해주고, Y좌표값은 B배로 해줌으로써 다음과 같은 타원을 얻을 수 있다.

$$\frac{X^2}{A^2}+\frac{Y^2}{B^2}=1$$

이러한 크기조절 행렬은 다음과 같다.

$$\begin{pmatrix}A & 0 & 0\\ 0 & B & 0\\ 0 & 0 & 1\end{pmatrix}$$

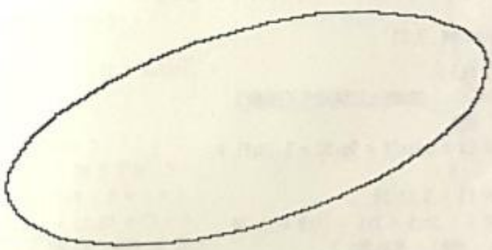
다음의 프로그램은 크기조절, 회전, 선형이동등을 적용시켜 타원을 그리게 된다.

* PROGRAM 2.26

```
1  REM
       ELLIPSE
2  REM
10 CX = 140:CY = 96:SC = 1.16:X =
    1:Y = 0:FL = 0
20 A = 90:B = 50
30 DT = .1:C =  COS (DT):S =  SIN
    (DT):N = 6.4 / DT
40 H = 10:K = 30:T = .3:S1 =  SIN
    (T):C1 =  COS (T)
50  HGR2 : HCOLOR= 3
60  FOR I = 1 TO N
70 T = X * C - Y * S:Y = Y * C +
    X * S:X = T
80 X1 = X * A * C1 - YB * S1 + H:
    Y1 = X * A * S1 + Y * B * C1
    + K
90 SX = SC * X1 + CX:SY = CY - Y1

100  IF SX < 0 OR SX > 279 OR SY <
    0 OR SY > 191 THEN FL = 0: GOTO
    140
110  IF FL = 1 THEN 130
120  HPLOT SX,SY:FL = 1
130  HPLOT  TO SX,SY
140  NEXT I
150  END
```

]



매개함수에 의한 곡선

다음과 같은 함수를 媒介函數(매개함수 : parametric equations)라고 한다.

$X = T^2 + 10$

$Y = T^2 - 10T$

이들은 X, Y값을 계산하기 위해 매개변수 T를 사용하고 있다.

과학, 공학 혹은 상업등에 대한 수학적인 응용들에서는 매개함수가 자주 쓰이는데 그것은 매개변수와 X, Y값이 응용에 있어 매우 중요한 의미를 갖기 때문이다. 우리는 X와 Y사이의 매개적인 함수관계를 주로 사용해 왔는데, 그 이유는 그것을 사용함으로써 보다 유연한 곡선을 짧은 연산 시간내에 구할 수 있기 때문이다.

다음의 프로그램 「DRAWS LISSAJOUS FIGURES」은 다음과 같은 매개함수로 정의된 그래프를 그리는 것이다.

$$X = 50\ SIN(2\ (T - \frac{\pi}{13}))$$

$$Y = -70\ COS\ (T)$$

이 매개함수는 다음과 같은 일반적인 형태의 매개함수의 특수한 형태이다.

$$X = A\ SIN\ (B\ (T - \frac{\pi}{C}))$$

$$Y = D\ COS\ (T)$$

예로서 A=50, B=2, C=14, D=−70이라고 한다면 어떤 곡선이 될까? 여기에서 A, B, C, D 값을 바꾸어 다른 곡선을 그려보아라.

다음과 같은 매개함수를 그려보면 매우 흥미로울 것이다.

$$\begin{cases} X = 30T^2 \\ Y = 30T^3, \quad -2 \leq T \leq 2 \end{cases}$$

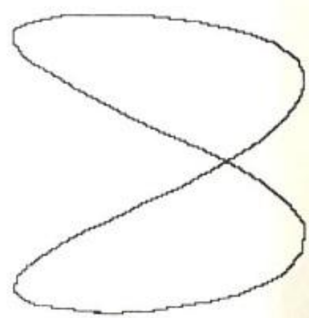
$$\begin{cases} X = 80\ COS\ T \\ Y = 80\ SIN\ T, \quad 0 \leq T \leq 2\pi \end{cases}$$

$$\begin{cases} X = 30\ T - 50\ SIN\ T \\ Y = 30 - COS\ T, \quad -4 \leq T \leq 4 \end{cases}$$

$$\begin{cases} X = 5\ (COS\ T + T SIN\ T) \\ Y = 5\ (SIN\ T - T COS\ T), \quad 0 \leq T \leq 20 \end{cases}$$

* PROGRAM 2.27

```
1  REM
        DRAWS LISSAJOUS FIGURES
2  REM
10 CX = 140:CY = 96:SC = 1.16:FL =
     0
20 PI = 3.14159
30 A = 50:B = 2:C = 14:D =  - 70
40  HGR2 : HCOLOR= 3
50  FOR T = 0 TO 6.3 STEP .1
60 X = A *  SIN (B * (T - PI / C)
     ):Y = D *  COS (T)
70 SX = SC * X + CX:SY = CY - Y
80  IF SX < 0 OR SX > 279 OR SY <
     0 OR SY > 191 THEN FL = 0: GOTO
     110
90  IF FL = 0 THEN  HPLOT SX,SY:F
     L = 1
100  HPLOT  TO SX,SY
110  NEXT T
```



포물선

포물선은 많은 면에서 실제적으로 응용되는 곡선이다. 예를 들어, 포탄이나 돌을 공중에 던지면, 그것이 지나간 자취는 포물선이 된다. 즉 바람이나 공기의 저

항이 없다고 가정하면 이들은 포물선 운동을 한다.

식 $Y=X^2/4P$는 그림 2-19와 같은 포물선을 나타내는 것이다.

원점은 포물선의 꼭지점이 된다. 그리고 초점은 점(0, P)에 있다. 포물선에서의 초점과 꼭지점을 지나는 직선을 그 포물선의 축이라고 한다. P값에 의해 초점과 꼭지점간의 상대적인 위치가 결정되고 또한 포물선의 모양이 결정된다. 초점을 지나면서 포물선의 축과 수직인 선분의 길이는 $4|P|$이다. 이 값에 의해 포물선의 넓이(width)가 결정된다.

방정식 $Y=X^2/4P$에 의해 포물선이 정의되지만, 이것보다 더 유용한 표현식으로 다음과 같은 매개함수가 있다.

$$X=2PT$$

$$Y=PT^2$$

이러한 식이 프로그램내에 있으면 당신은 P값을 바꾸어 봄으로써 여러가지 모양의 포물선들을 신속하게 그려볼 수 있을 것이다.

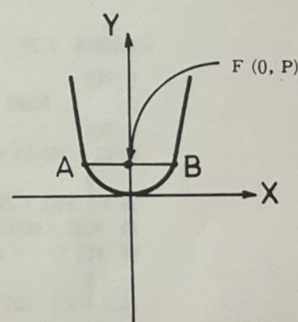
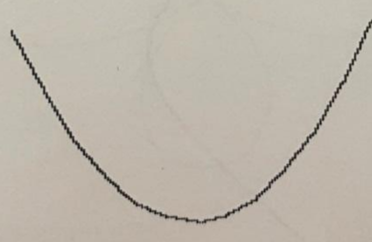


그림 2-19 포물선 $Y=X^2/4P$

```
* PROGRAM 2.28

 1  REM
        DRAWS A PARABOLA
 2  REM
 10 CX = 140:CY = 96:SC = 1.16:FL =
     0
 20 P = 10
 40  HGR2 : HCOLOR= 3
 50  FOR T =  - 5 TO 5 STEP .5
 60 X = 2 * P * T:Y = P * T * T
 70 SX = SC * X + CX:SY = CY - Y
 80  IF SX < 0 OR SX > 279 OR SY <
     0 OR SY > 191 THEN FL = 0: GOTO
     110
 90  IF FL = 0 THEN  HPLOT SX,SY:F
     L = 1
 100  HPLOT  TO SX,SY
 110  NEXT T

]
```



쌍곡선 원, 타원, 포물선 등을 원뿔의 단면곡선이라고 한다. 왜냐하면 이 곡선들은 원뿔과 평면과의 교차곡선들이기 때문이다. 쌍곡선도 원뿔의 단면 곡선들 중의 하나이다.

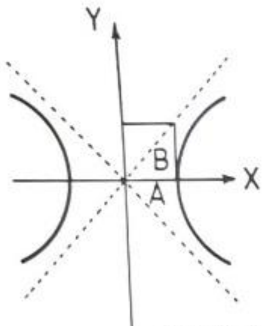


그림 2-20 쌍곡선

쌍곡선은 대수적으로 다음과 같이 표현된다.

$$\frac{X^2}{A^2}-\frac{Y^2}{B^2}=1$$

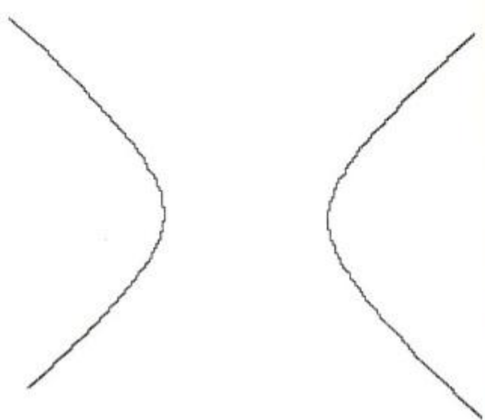
쌍곡선의 모양은 A, B의 값을 바꿈으로써 여러가지로 바뀐다. 쌍곡선의 매개 함수적 표현은 다음과 같으며 이것을 사용하면 보다 쉽게 프로그램을 작성할 수 있다.

$$X=A\,SEC(T)$$
$$Y=B\,TAN(T)$$

이러한 것을 다음 프로그램에서 나타내었다.

```
* PROGRAM  2.29
 1  REM
          DRAWS A HYPERBOLA
 2  REM
 10 CX = 140:CY = 96:SC = 1.16:FL =
     0
 20 A = 30:B = 35
 40  HGR2 : HCOLOR= 3
 50  FOR T =  - 1.6 TO 4.5 STEP .0
     5
 60 X = A /  COS (T):Y = B *  TAN
     (T)
 70 SX = SC * X + CX:SY = CY - Y
 80  IF SX < 0 OR SX > 279 OR SY <
     0 OR SY > 191 THEN FL = 0: GOTO
     110
 90  IF FL = 0 THEN  HPLOT SX,SY:F
     L = 1
 100  HPLOT  TO SX,SY
 110  NEXT T

   ]
```



매개함수의 변형 곡선 도형

반지름 R이 90인 원은 다음과 같은 매개함수로 표현할 수 있다.

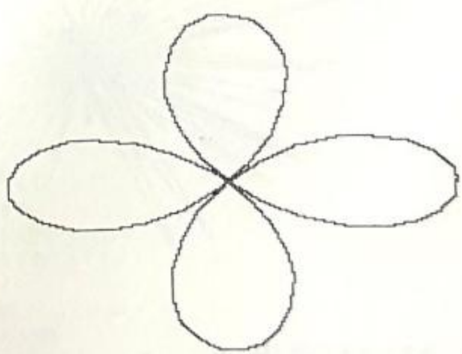
X = R COS(T)

Y = R SIN(T)

만약 여기서 R의 값을 일정하게 놓지 않고 여러가지로 변할 수 있게 한다면,즉 R을 변수로 사용하면 매우 재미있는 곡선들을 얻을 수 있다. R을 T에 대한 함수로 정의하여 다음의 프로그램을 작성하였다.

```
* PROGRAM  2.30
 1  REM
       DRAWS A POLAR COORDINATE
        CURVES
 2  REM
 10 CX = 140:CY = 96:SC = 1.16:FL =
     0
 40  HGR2 : HCOLOR= 3
 50  FOR T = 0 TO 6.3 STEP .1
 55 R = 80 *  COS (2 * T)
 60 X = R *  COS (T):Y = R *  SIN
     (T)
 70 SX = SC * X + CX:SY = CY - Y
 80  IF SX < 0 OR SX > 279 OR SY <
     0 OR SY > 191 THEN FL = 0: GOTO
     110
 90  IF FL = 0 THEN  HPLOT SX,SY:F
     L = 1
 100  HPLOT  TO SX,SY
 110  NEXT T

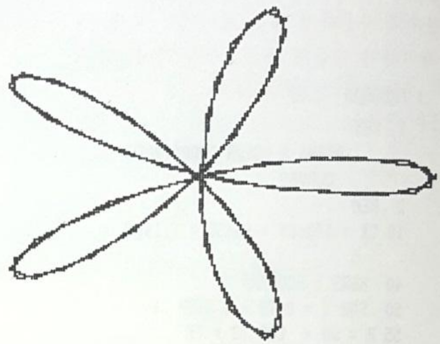
 ]
```



여기서 행번호 55를 다음과 같이 수정하면,

```
55 R = 80 * COS (5 * T)
```

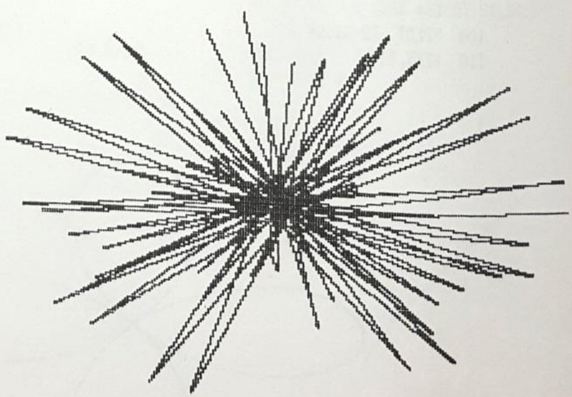
변수 R의 변화에 의해 다음과 같은 수행결과를 얻는다.



또 행번호 55를

```
55 R = 100 * COS (100 * T)
```

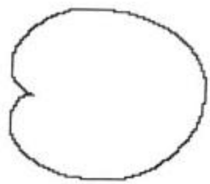
으로 수정하여 수행시키면 다음과 같은 결과가 나타난다.



또,

```
55 R = 30 * ( COS (T) - 1)
```

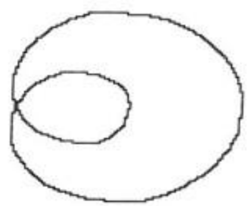
으로 해서 수행시키면 다음과 같다.



마찬가지로

```
55 R = 20 * (3 *  COS (T) - 1)
```

로 수정시켜 보자.



이렇게 R의 값을 변화시켜 프로그램을 작성함으로써 많은 화상변화를 가져다 준다. 당신도 변수 R에 대해 연구하기 바란다.

고리(RING)

지금까지의 프로그램을 이용하여 나선형처럼 그려지는 둥근 고리 모양을 그리는 프로그램을 작성해 보자.

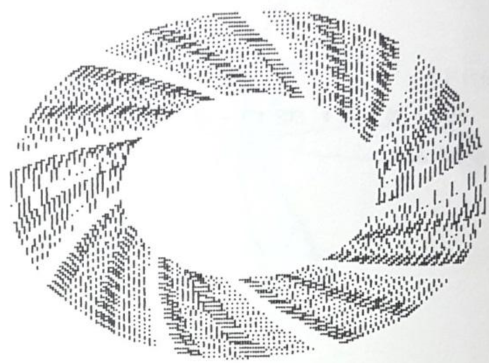
```
* PROGRAM  2.31

 1  REM
         RING
 2  REM
 10 DT = .1:C =  COS (DT):S =  SIN
      (DT)
 20 SC = 1.16:CX = 140:CY = 96
 30  HGR2 : HCOLOR= 3
 40 X = 0:Y = 80
 50  HPLOT SC * X + CX,Y + CY
 60  FOR J = 1 TO 20
 70  FOR I = 1 TO 63
 80 T = X * C - Y * S:Y = Y * C +
      X * S:X = T
 90  READ CL: POKE 28,CL: IF CL =
      255 THEN  RESTORE
 100  HPLOT  TO SC * X + CX,Y + CY

 110  NEXT I
```

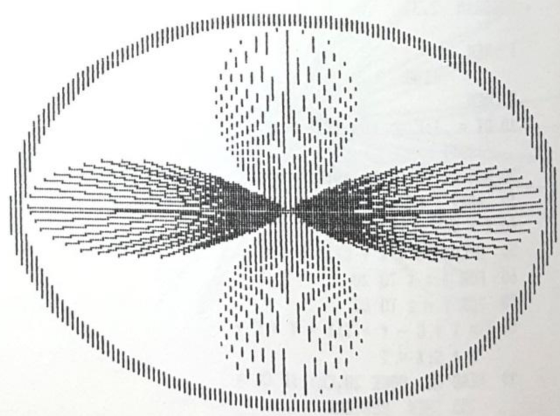
```
120 Y = Y - 2: NEXT J
130  DATA  42,85,127,128,170,213,
     255

]
```



문제 126. 극좌표 방정식 R=COS(2θ) 에 의해 구성되는 다음의 그림을 순환방법에 의해 계산하여 프로그램 을 작성하라.

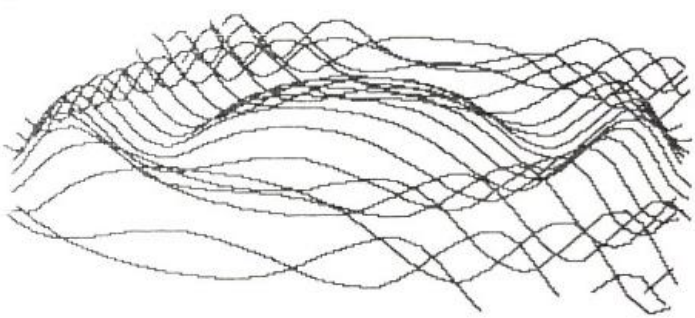
]



```
1  REM ***********************
2  REM *  POLAR  HEX  SIGN   *
3  REM ***********************
4  REM
10  HGR2 :SC = 1.16
20 CX = 140:CY = 96
30 C =  COS (.1):S =  SIN (.1)
40  REM  POLOR "ROSE"
50  FOR TH = 0 TO 6.3 STEP .05
60 R = 88 *  COS (2 * TH)
70  HCOLOR= 7: IF R < 0 THEN  HCOLOR=
     5
80 X = CX + SC * R *  COS (TH)
90 Y = CY + R *  SIN (TH)
100  HPLOT 140,96 TO X,Y
110  NEXT TH
120  REM  BOUNDARY CIRCLES
130  HCOLOR= 6
140  FOR R = 90 TO 95 STEP .8:X =
     R:Y = 0
150  FOR I = 0 TO 63
160 T = X * C - Y * S
170 Y = X * S + Y * C:X = T
180  IF I = 0 THEN  HPLOT CX + SC
     * X,CY - Y
190  HPLOT  TO CX + SC * X,CY - Y

200  NEXT I,R
```

문제 127. 3차원 물체인 다음 화상을 그리는 프로그램을 작성하라.



3차원 물체에 대한 화상은 곡면에 대한 기법을 이용하면 된다.

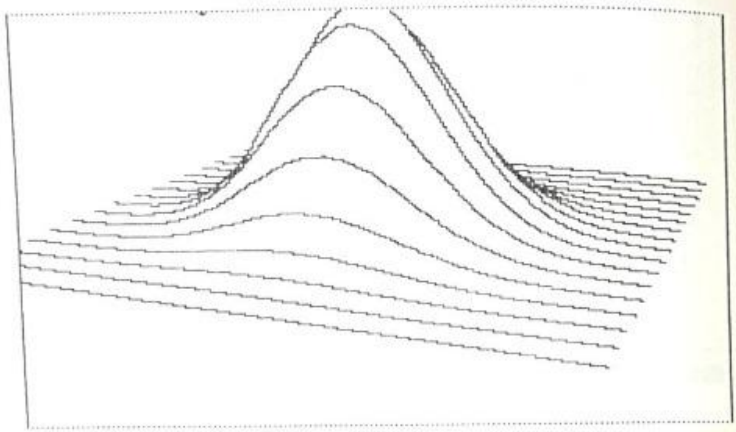
1. 곡면 위의 점들에 대한 직각좌표(X, Y, Z)를 구한다.
2. 이것들을 화면좌표(SX, SY)로 변환한다.
3. 이 점들을 그린다.

```
1  REM **********************
2  REM *  C  R  A  T  E  R  *
3  REM **********************
4  REM
10 Z = F(X,Y)
20  HGR2 : HCOLOR= 3:RHO = 30:THE
     TA = .4:PHI = 1.25:D = 420:C
     X = 140:CY = 96
30 S1 =  SIN (THETA):C1 =  COS (T
     HETA)
40 S2 =  SIN (PHI):C2 =  COS (PHI
     )
50  DEF  FN F(X) =  COS (.1 * (X *
     X + Y * Y))
60  REM  POINT GENERATING LOOP
70  FOR X = 10 TO  - 10 STEP  - 1

80 FL = 0
90  FOR Y =  - 10 TO 10 STEP .4
100 Z =  FN F(X)
110  GOSUB 150
120  NEXT Y,X
130  END
140  REM  PLOTTING SUBROUTINE
150 XE =  - X * S1 + Y * C1
160 YE =  - X * C1 * C2 - Y * S1 *
     C2 + Z * S2
170 ZE =  - X * S2 * C1 - Y * S2 *
     S1 - Z * C2 + RHO
180 SX = D * (XE / ZE) + CX
190 SY =  - D * (YE / ZE) + CY
200  IF SX < 0 OR SX > 279 OR SY <
     0 OR SY > 191 THEN FL = 0: RETURN

210  IF FL = 0 THEN  HPLOT SX,SY:
     FL = 1
220  HPLOT  TO SX,SY
230  RETURN
```

문제 128. 다음 화상을 그리는 프로그램을 작성하라.



```
1  REM ********************
2  REM * M O U N T A I N  *
3  REM ********************
4  REM
20  HGR2 : HCOLOR= 3:RHO = 30:THE
     TA = .4:PHI = 1.25:D = 420:C
     X = 140:CY = 96
30 S1 =  SIN (THETA):C1 =  COS (T
     HETA)
40 S2 =  SIN (PHI):C2 =  COS (PHI
     )
50  DEF  FN Z(X) = 7 *  EXP ( - .
     1 * (X ^ 2 + Y ^ 2))
60  DIM YN(280),YX(280)
70  FOR I = 0 TO 279:YN(I) = 191:
      NEXT I
80  REM  POINT GENERATING LOOP
90  FOR X = 8 TO  - 8 STEP  - 1
100 FL = 0
110  FOR Y =  - 8 TO 8 STEP .5
120 Z =  FN Z(X)
130  GOSUB 190
140  NEXT Y,X
150  HCOLOR= 2
160  HPLOT 0,0 TO 278,0 TO 278,19
     1 TO 0,191 TO 0,0
170  END
180  REM   PLOTTING SUBROUTINE
190 XE =  - X * S1 + Y * C1
200 YE =  - X * C1 * C2 - Y * S1 *
     C2 + Z * S2
210 ZE =  - X * S2 * C1 - Y * S2 *
     S1 - Z * C2 + RHO
220 SX = D * XE / ZE + CX
230 SY =  - D * YE / ZE + CY
240  IF FL = 0 THEN FL = 1:F = 0:
      GOTO 360
250 DX = OX - SX: IF DX = 0 THEN
      DX = 1
260 SL = (OY - SY) / DX:YP = OY
270  FOR XP =  INT (OX) + 1 TO SX

280 FG = 1
290 YP = YP + SL
300  IF XP < 0 OR XP > 279 THEN F
      G = 0:F = 0: GOTO 350
310  IF YP > 191 OR YP < 0 THEN F
      G = 0:F = 0
320  IF YP <  = YN(XP) THEN 370
330  IF YP >  = YX(XP) THEN 420
340 F = 0
350  NEXT XP
360 OX = SX:OY = SY: RETURN
370 YN(XP) = YP
```

```
380  IF FG = 0 THEN 410
390  IF F = 0 THEN  HPLOT XP,YP:F
     = 1
400  HPLOT  TO XP,YP
410  IF YP < YX(XP) THEN 350
420 YX(XP) = YP
430  IF FG = 0 THEN 460
440  IF F = 0 THEN  HPLOT XP,YP:F
     = 1
450  HPLOT  TO XP,YP
460  GOTO 350
```

2-5 화면절환 및 기타 그래픽의 기법

화면절환

여러가지 모드, 페이지, 혼합 중에서 선택하는 디바이스를 「**소프트스위치** (soft switch)」라고 한다. 이것은 컴퓨터의 소프트웨어로 제어된다. 프로그램에서는 스위치를 위한 특별한 메모리 번지를 참조하는 것으로 스위치를 넣을 수 있다. 그 번지에서 해득하거나 써 넣거나 하는 데이타에는 무관하다.

화면이 소프트 스위치의 설정을 제어하는 특별한 메모리 번지가 8개 있다. 그것은 각각 두개가 1세트로 되어 있으므로, 다음과 같은 4개의 스위치로 생각하면 편리하다.

1. 텍스트-그래픽 스위치
2. 전체화면-혼합화면 스위치
3. 페이지1-페이지2 스위치
4. 고해상도-저해상도 스위치

각 스위치는 화면에 출력하는데 관련된, 특정한 모드만을 제어하는데, BASIC 명령문을 이용하여 한 번에 여러개의 스위치를 제어할 수 있다. 예를 들어 HGR 명령은 스위치 1을 그래픽모드로, 스위치 2를 혼합화면 모드로, 스위치 3을 페

표 2-5 POKE 명령

스위치	동 작	메모리 번지		BASIC 명령
		16진수	10진수	
1	그래픽스모드	$ C050	49232	POKE-16304, 0
	텍 스 트모드	$ C051	49233	POKE-16303, 0
2	전체화면모드	$ C052	49234	POKE-16302, 0
	혼합화면모드*	$ C053	49235	POKE-16301, 0
3	페 이 지 1	$ C054	49236	POKE-16300, 0
	페 이 지 2	$ C055	49237	POKE-16299, 0
4	저 해 상 도*	$ C056	49238	POKE-16298, 0
	고 해 상 도*	$ C057	49239	POKE-16297, 0

* 이들 모드는 "그래픽스의 표시" 스위치가 ON인때만 보인다.

이지 1로, 그리고 스위치 4를 고해상도로 지정한다.

이들 4개의 스위치에 대한 독립적인 제어는 표2-5와 같이 POKE명령을 사용함으로써 가능하다.

각 스위치들은 두 개의 선택모드 중의 어느 하나로 선택되는데, 절환 스위치로 지정된다. 일단 지정되면, 그 스위치가 다른 모드로 변경되기 전에는 그대로 남아 있게 된다.

4개의 스위치가 독자적으로 제어될 수 있기 때문에 16가지의 디스플레이 모드가 가능할 것 같으나 실제로는 그렇지 않다.

스위치 1이 텍스트모드(POKE-16303, 0)로 지정되어 있으면 스위치 2와 4는 아무런 효과도 얻지 못하므로 실제적으로 10개의 서로 다른 디스플레이 모드만이 의미를 갖게 된다. 어떤 한 디스플레이 모드에서 다른 디스플레이 모드로 바꾸려면 관련된 스위치만 바꾸면 된다.

이러한 스위치의 가능한 표시방식은 다음과 같다.

표 2-6 가능한 디스플레이 방식과 스위치 명령

모 드	페 이 지 1		페 이 지 2	
	스 위 치	BASIC 명령	스 위 치	BASIC 명령
전 체 텍 스 트	$C054 $C051	POKE -16300,0 : POKE -16303,0	$C055 $C051	POKE -16299,0 : POKE -16303,0
전 체 화 면	$C054 $C056	POKE -16300,0 : POKE -16298,0	$C055 $C056	POKE -16299,0 : POKE -16298,0
저해상도그래픽스	$C052 $C056	POKE -16302,0 : POKE -16298,0	$C052 $C050	POKE -16302,0 : POKE -16298,0
전 체 화 면	$C054 $C057	POKE -16300,0 : POKE -16297,0	$C055 $C057	POKE -16299,0 : POKE -16297,0
고해상도그래픽스	$C052 $C050	POKE -16302,0 : POKE -16304,0	$C052 $C050	POKE -16302,0 : POKE -16304,0
혼 합 화 면	$C054 $C056	POKE -16300,0 : POKE -16298,0	$C055 $C056	POKE -16299,0 : POKE -16298,0
저해상도그래픽스	$C053 $C050	POKE -16301,0 : POKE -16304,0	$C053 $C050	POKE -16301,0 : POKE -16304,0
혼 합 화 면	$C054 $C057	POKE -16300,0 : POKE -16297,0	$C055 $C057	POKE -16299,0 : POKE -16297,0
고해상도그래픽스	$C053 $C050	POKE -16301,0 : POKE -16304,0	$C053 $C050	POKE -16301,0 : POKE -16304,0

소프트 스위치로 디스플레이 모드를 제어할 수는 있지만 화상을 그리려고 하는

그래픽 페이지 자체에 대해서는 제어할 수 없다.

메모리 번지 $ 00E6 (230) 은 화상을 그리려고 하는 그래픽 페이지를 식별하는데, 만약 페이지 1 이 사용되려면, 이 메모리 번지에는 $ 0020 (32) 가 지정되어야 하고, 만약 페이지 2 가 사용되려면 $ 0040 (64) 가 기억되어야 한다. 메모리 번지 $ 00E6 (230) 의 내용은 디스플레이 모드와는 독립적으로 제어되기 때문에 어느한 페이지가 디스플레이되고 있는 동안, 다른 페이지를 다음에 디스플레이 할 화상을 그리는데 사용할 수 있다.

다음의 프로그램은 어느 한 페이지가 디스플레이 되는 동안, 다른 페이지에 화상을 그리는 과정을 설명하기 위한 것이다. 이것을 실행시켜 보자.

* PROGRAM 2.32

```
1  REM  ILLUSTRATES ANIMATION TEC
     HNIQUES

10  HGR : HGR2
20 A = 1:B =  - 1:C = 2:D = 1
30  FOR I = 1 TO 20
40  FOR J = 1 TO 2
50  POKE 230,32 * J
60 A = A + B
70  HCOLOR= 0
80  HPLOT 140 - A,96 - A TO 140 +
     A,96 - A TO 140 + A,96 + A TO
     140 - A,96 + A TO 140 - A,96
     - A
90 A = A + C
100  HCOLOR= 3
110  HPLOT 140 - A,96 - A TO 140 +
     A,96 - A TO 140 + A,96 + A TO
     140 - A,96 + A TO 140 - A,96
     - A
120  POKE  - 16299 - D,0
130 D = 1 - D
140  NEXT J,I
150 B =  - B:C =  - C
160  GOTO 30
```

이 프로그램을 수행시켜 보면 사각형이 점점 확대되었다가 축소되는 과정이 반복된다. 이 프로그램을 중단시키려면 CTRL-C 나 RESET 키를 누르면 된다.

행번호 50에서는 그림을 그리려고 하는 페이지를 지정한다. 여기에서 화상은 디스플레이 되고 있지않은 페이지에서 항상 그려지고 있음에 유의하여야 한다. 행 60에서 행 80까지는 바탕색인 검은색으로 화상을 그림으로써 실제로는 그래픽 페이지에 앞서 그려졌던 화상을 지우는 효과를 나타낸다. 행 90에서 110까지는 조금 더 확대되었거나 혹은 축소된 새로운 사각형을 그리는 곳이다. 행번호 120에서는 새로운 사각형을 디스플레이 하기 위해 페이지를 전환하고 있다. 이 프로그램은 디스플레이 되고 있지 않은 페이지의 화상을 지우고 다시 그리는 과정을 계속 반복한다.

메모리이동

이상과 같이 애플 컴퓨터를 이들 모드의 어느 것에 설정하는데는, 그 모드를 설정하는 스위치에 대응한 메모리 번지를 참조하기만 하면 된다. 기계어 프로그램에서는 상술의 16진 메모리 번지를 사용해야 하지만, 애플 소프트 BASIC프로그램에서는 상응하는 10진 메모리 번지를 PEEK 또는 POKE 하지 않으면 안된다. 그래픽 모드의 어느 것에 스위치할 때에 스위치는 어떤 순번으로 넣어도 되는데, 그래픽 모드의 어느 것에 스위치할 때에는 텍스트 그래픽의 스위치를 최후에 넣으면 될 것이다. 그 이외의 모드의 변경

은 텍스트의 그늘에 숨어 있어 볼 수 없으므로, 그래픽 모드에 설정했을 때, 마무리된 화면이 한번에 나타난다.

PEEK와 POKE에 의해 메모리의 어느 한 부분으로 부터 다른 부분으로 데이타를 옮기는 것에 대해 살펴보자.

여기서 PEEK는 지정된 메모리 번지의 내용을 읽는 것이고, POKE는 지정된 메모리 번지에 어떤 값을 새로 써 넣는 것이다. 따라서

```
]POKE 9,PEEK( 8)
```

이란 명령문은 번지 8 (PEEK (8))의 내용을 읽어서, 메모리 번지 9에 저장하는 역할을 수행한다. 그러나 메모리 번지 8의 내용은 영향을 받지 않는다.

PEEK와 POKE 명령을 써서 어떤 메모리 부분의 내용을 다른 부분으로 옮길 수 있는데, 예를 들면, 다음의 프로그램은 고해상도 그래픽 페이지 2의 내용을 고해상도 그래픽 페이지 1로 옮겨준다.

```
* PROGRAM  2.33
  10  FOR I = 8192 TO 16383
  20  POKE I, PEEK (8192 + I)
  30  NEXT I
        :
        :
```

그러나 위의 프로그램은 너무 느려 실제 응용에는 부적당하다. 다행히 상기 프로그램과 같은 역할로 훨씬 빠른 기계어 레벨의 프로그램을 사용할 수 있는데, 다음과 같이 먼저 초기화시켜 줌으로써 당신은 BASIC 프로그램내에서 기계어 레벨의 프로그램을 호출하여 사용할 수 있다.

```
POKE 768,160:POKE 769,0
POKE 770,76:POKE 771,44
POKE 772,254
```

그리고 나서 어떤 메모리의 내용을 새로운 부분으로 옮기고자 할 때마다 다음 사항을 지정해 주어야 한다.

1. 이동될 부분의 시작번지
2. 이동될 부분의 끝번지
3. 목적지 부분의 시작번지

마지막으로 「CALL768」을 하면 기계어 레벨의 이동 프로그램이 수행된다.

각각의 3개의 번지들은 두 단계로 주어져야 한다. 즉 각 번지의 상위 바이트와 하위 바이트는 이동 프로그램에 의해 사용될 수 있게 하기 위해 표2-7과같이 저장되어야 한다.

예를 들면 다음 프로그램은 이동 프로그램을 이용하여 고해상도 그래픽 페이지 2의 내용을 고해상도 그래픽 페이지 1에 옮긴 후, 메모리 번지 $6000부터 $

표2-7

번 지	내 용
60	이동될 부분의 시작 번지의 하위바이트
61	이동될 부분의 시작 번지의 상위바이트
62	이동될 부분의 끝 번지의 하위바이트
63	이동될 부분의 끝 번지의 상위바이트
66	목적지 부분의 시작 번지의 하위바이트
67	목적지 부분의 시작 번지의 상위바이트

7FFF까지의 내용을 고해상도 그래픽 페이지 2에다 옮기는 작업을 수행한다.

* PROGRAM 2.34

```
10  REM  MEMORY MOVE SUBROUTINE
20  POKE 768,160: POKE 769,0
30  POKE 770,76: POKE 771,44
40  POKE 772,254
50  REM  PAGE 2 TO PAGE 1
60  POKE 60,0: POKE 61,64
70  POKE 62,255: POKE 63,95
80  POKE 66,0: POKE 67,32
90  CALL 768
100  REM  PAGE 3 TO PAGE 2
110  POKE 60,0: POKE 61,96
120  POKE 62,255: POKE 63,127
130  POKE 66,0: POKE 67,64
140  CALL 768
```

PEEK와 POKE를 이용하여 데이타를 옮기는 루틴은 BASIC프로그램 내에서 매우 간단히 처리되지만, 기계어 레벨의 이동 프로그램을 초기화시킨 후 호출하여 사용하는 방법이 훨씬 빨리 수행된다.

이동 프로그램을 사용할 때는 반드시 주의해야 할 사항이 있다. 만약 목적지 부분의 시작번지가 이동될 부분의 중간에 놓이면 그 윗부분은 복사되어지기 전에 이동될 부분의 앞부분의 내용으로 대치되기 때문에 원하는 결과를 얻을 수 없다. 이러한 것은 일반적으로 배제되어야 하지만, 만약에 의도적으로 어떤 부분을 복사하고 싶을 때는 이동될 부분의 내용을 다음의 프로그램과 같이 중복시킴으로써 얻을 수도 있다.

* PROGRAM 2.35

```
1  REM  NAME COPIER
10  POKE 768,160: POKE 769,0
20  POKE 770,76: POKE 771,44
30  POKE 772,254
40  TEXT : HOME
```

```
50  INPUT "YOUR NAME, PLEASE ? ";
      A$
60  HOME : PRINT A$
70 L =  LEN (A$) + 1
80  POKE 60,0: POKE 61,4
90  POKE 62,255 - L: POKE 63,7
100  POKE 66,L: POKE 67,4
110  CALL 768: WAIT  - 16384,128
120  HOME
```

기타 그래픽 페이지의 구성방법

앞의 프로그램「MEMORY MOVE SUBROUTINE」에서 메모리 번지 $6000에서 $7FFF까지의 내용을 고해상도 그래픽 페이지 2에 옮기는 과정이 있다. 이러한 일은 만약 고해상도의 화상이 $6000에서 $7FFF사이에 기억되어 있지 않으면 구태여 할 필요가 없다. 메모리 번지 $6000에서 $7FFF 사이에 화상을 넣어 두는 방법은,

첫째, 디스크로 부터 화상을 가져온다.

```
BLOAD PICTURE, A$6000
```

둘째, 화상을 고해상도 페이지 1이나 2에 그린 후 그것을 $6000에서 $7FFF사이에 복사한다.

셋째, 고해상도 페이지 3 ($6000 - $7FFF)에 직접 그린다.

셋째 방법은 BASIC에서 쉽게 사용할 수 있다. 먼저 화상이 그려질 영역으로서 페이지 3을 지정하고,

```
POKE 230, 96
```

그곳의 내용을 흑백으로 모두 지운 다음

```
CALL 62450
```

정상적인 그래픽 명령들로서 화상을 그리면 된다. 그래픽 페이지 4로써 메모리 번지 $8000에서 $9FFF까지가 사용될 수 있다.

그러나 이 부분은 DOS 프로그램이 차지하고 있기 때문에, 이곳의 내용을 모두 지우고 화상을 그리게 되면 DOS 프로그램의 일부가 지워져 그 기능을 못하게 된다. 그러므로 프로그램이 끝난 후에는 다시 시스템을 초기화시켜야 한다. 즉 DOS프로그램을 다시 디스크로 부터 메모리에 가져와야 한다.

고해상도 그래픽 페이지 5로서 메모리 번지 $A000에서 $BFFF까지를 페이지 4와 같은 방법으로 사용할 수 있다.

페이지 4나 5를 사용할 때는 문자열들을 그곳에 기억시키는 것을 피해야 한다. 문자열들은 보통 HIMEM 바로 아래에 기억되므로 프로그램에서는 미리 HIMEM을 8192로 지정해 줄 필요가 있다.

```
HIMEM: 8192
```

이렇게 함으로써 문자열들은 고해상도 그래픽 페이지 1의 바로 아래 부분에 기억된다.

다음의 프로그램은 고해상도 그래픽 페이지 1과 2 이외에 다른 메모리 번지에 화상을 그리기 위해 소프트 스위치들을 사용하는 방법이 설명되고 있다.

* PROGRAM 2.36

```
1  REM  ADDITIONAL GRAPHICS PAGES
     (DRAWS,DISPLAYS 4 PAGES OF
     GRAPHICS)
10  TEXT : HOME : HCOLOR= 3
20  POKE 768,160: POKE 769,0
30  POKE 770,76: POKE 771,44
40  POKE 772,254
50  POKE 230,64: CALL 62450
60  HPLOT 140,10 TO 140,180
70  POKE 230,96: CALL 62450
80  HPLOT 120,10 TO 120,180
90  HPLOT 160,10 TO 160,180
100  IN# 0: PR# 0
110  POKE 230,128: CALL 62450
120  HPLOT 110,10 TO 110,180
130  HPLOT 140,10 TO 140,180
140  HPLOT 170,10 TO 170,180
150  POKE 230,160: CALL 62450
160  HPLOT 110,10 TO 110,180
170  HPLOT 170,100 TO 10,100 TO 1
     10,10
180  PRINT "THIS PROGRAM WILL DIS
     PLAY 1 OF 4": PRINT
190  PRINT "HIGH RESOLUTION IMAGE
     S": PRINT
200  PRINT "PRESS 1,2,3 OR 4 TO S
     EE THEM": PRINT
210  PRINT "PRESS (ESC) TO END TH
     E PROGRAM": PRINT
220  PRINT "PRESS (RETURN) TO RET
     URN TO THIS MENU"
230  POKE - 16297,0: POKE - 163
     02,0
240  VTAB 18: GET A$: PRINT :X =
     FRE (0)
250  IF ASC (A$) = 27 THEN TEXT
     : HOME : PRINT "DOS IS DEAD!
     ": END
260  IF ASC (A$) = 13 THEN POKE
     - 16303,0: GOTO 240
270 A = VAL (A$): ON A GOTO 290,
     340,390,440
280  GOTO 240
290  POKE - 16304,0
300  POKE 60,0: POKE 61,64
310  POKE 62,255: POKE 63,95
320  POKE 66,0: POKE 67,32
330  CALL 768: GOTO 240
340  POKE - 16304,0
350  POKE 60,0: POKE 61,96
360  POKE 62,255: POKE 63,127
370  POKE 66,0: POKE 67,32
380  CALL 768: GOTO 240
390  POKE - 16304,0
400  POKE 60,0: POKE 61,128
410  POKE 62,255: POKE 63,159
420  POKE 66,0: POKE 67,32
430  CALL 768: GOTO 240
440  POKE - 16304,0
450  POKE 60,0: POKE 61,160
460  POKE 62,255: POKE 63,191
470  POKE 66,0: POKE 67,32
480  CALL 768: GOTO 240
```

이 프로그램을 실행시켜 보면, 행번호 250에서는 DOS가 동작 불능으로 되면서 끝나게 된다. 행번호 250을 다음과 같이 수정하면,

```
250 IF ASC(A$) = 27 THEN TEXT: CALL 50688
```

디스크 콘트롤러(controller)가 슬롯 6에 있을 경우 다시 DOS가 메모리에 옮겨진 후 끝나게 된다.

TEXT 페이지 2 애플에서는 POKE-16303, 0와 POKE-16299, 0를 지정해 줌으로써 텍스트 페이지 2의 내용을 디스플레이 시킬 수 있다. 그러나 이 페이지에 어떤 내용을 써 넣는 것은 쉽지 않다. 우리의 주된 관심은 고해상도 그래픽이지만 텍스트 페이지 2의 내용을 디스플레이 시켜 볼 필요가 있다. 특히 고해상도 혼합화면모드의 그래픽에서 화면의 밑부분에 나타나는 텍스트는 텍스트 페이지 2로부터 가져온 것이다.

텍스트 페이지 2를 사용하는데 있어서의 첫번째 문제점은 그것이 보통 애플소프트 BASIC 프로그램이 사용하는 메모리 번지 $0800에서 $0BFF에 위치하고 있다는 것이다. BASIC 프로그램 내에서 텍스트 페이지 2를 사용하려면, 프로그램을 다른 메모리 영역에 기억시켜야 한다. 이것은 쉽게 처리할 수 있다. 메모리 번지 $0067(103)과 $0068(104)에는 실제의 애플 소프트 프로그램의 시작번지의 하위바이트와 상위바이트가 각각 들어 있다. 보통 메모리 번지 103에는 $01(1), 메모리 번지 104에는 $08(8)이 들어 있다. 실제로 프로그램을 기억시키기 위해 마련된 메모리 번지는 $800 부터이지만 메모리 번지 $800에는 0이 들어가고 실제의 프로그램은 $801에서 부터 시작된다.

만약 어떤 BASIC 프로그램이 텍스트 페이지 1,2와 고해상도 그래픽 페이지1, 2를 사용하고 있다면, 프로그램을 기억시키기 위한 적당한 큰 메모리 블록은 $6000(24576)에서 부터 시작된다.

이곳을 프로그램 영역으로 사용하려면 메모리 번지 103과 104에는 각각 $01(1)과 $60(96)이 지정되어야 한다. (24576=96×256)

아울러 메모리 번지 $6000에는 프로그램의 시작 코드인 0을 지정해 주어야 한다. 이곳에 프로그램을 써 넣으려면 다음과 같이 한 후 작성한다.

```
]POKE 103,1

]POKE 104,96

]POKE 24576,0

]NEW
```

만약 디스크로 부터 프로그램을 이 곳으로 가져와서 수행시키려면 다음과 같이 한다.

```
]POKE 103,1

]POKE 104,96

]POKE 24576,0

]RUN PROGRAM
```

위의 4행은 키보드에서 받아들일 수도 있고, 또는 애플 소프트 프로그램의

일부일 수도 있으며, 텍스트 화일로 부터 가져와서 수행될 수도 있다.

BASIC 프로그램을 텍스트 페이지 2로부터 분리시켜 다른 메모리 번지에 저장시키는 방법을 알았으므로 이제는 텍스트 페이지 2에 텍스트를 써 넣는 방법을 다루어 보자.

첫째, 정상적인 방법으로 텍스트 페이지 1에 텍스트를 써 넣은 후, 기계어 레벨의 이동프로그램을 이용하여 텍스트 페이지 1의 내용을 텍스트 페이지 2로 옮긴다.

둘째, 직접 텍스트 페이지 2에 써 넣는다.

두번째 방법을 사용하기 위해서는 먼저 텍스트 페이지 2를 텍스트를 써 넣을 장소로 지정해 주어야 한다. 정상상태에서는 메모리번지 $29(41)에 4에서 7사이의 숫자가 저장되어 있어 $0400에서 $07FF까지의 1024개의 메모리번지들을 텍스트나 저해상도 그래픽을 기억시킬 장소로 저장하게 된다.

따라서 $29에 8, 9, 10, 11중의 어떤 값을 써 넣음으로써 $0800에서 $0BFF까지의 메모리 번지를 지정해 줄 수 있다. 이렇게 함으로써 텍스트 페이지 2에 직접 텍스트를 써 넣을 수 있다. 그러나 불행하게도 메모리 번지 $29의 내용은 BASIC프로그램의 각 문장끝에 항상 들어가는 RETURN을 만날 때마다 4에서 7사이의 값으로 바뀌게 되어 있다. 그러므로 텍스트 페이지 2에 어떤 내용을 써 넣기 바로전에 메모리번지 $29의 내용을 수정해 주는 명령을 반드시 써 주어야 한다. 다음의 프로그램은 이러한 것들을 잘 설명해 준다. 이를 실행시켜 보자.

* PROGRAM 2.37

```
1  REM  PAGE 2 OF TEXT (ILLUSTRAT
      ES PRINTING, DISPLAYING PAGE
       TWO OF TEXT)
5  REM  GREETING-2 PROGRAM
10  HOME : VTAB 10
20  FOR I = 2048 TO 3071: POKE I,
     160: NEXT I
30  POKE  - 16299,0
40  SPEED= 55
50  POKE 41, PEEK (41) + 4: PRINT
     "WE ARE NOW PRINTING ON PAGE
     2."
60  POKE 41, PEEK (41) + 4: PRINT
     "GET READY FOR A CHANGE TO P
     AGE 1."
70  FOR I = 1 TO 1000: NEXT I
80  POKE  - 16300,0: VTAB 10
90  PRINT "WE ARE NOW PRINTING ON
     PAGE 1."
100  PRINT "WE WILL DISPLAY EACH
     OF THE TWO"
110  PRINT "TEXT PAGES REPEATEDLY
     "
120  PRINT "WHILE COUNTING FROM 1
      TO 40"
130  SPEED= 255
140  FOR J = 1 TO 20
150  POKE  - 16299,0
160  VTAB 18: POKE 41, PEEK (41) +
     4: PRINT 2 * J - 1
170  FOR I = 1 TO 1000: NEXT I
180  VTAB 18: PRINT 2 * J
190  POKE  - 16300,0
200  FOR I = 1 TO 1000: NEXT I
210  NEXT J
220  END
```

반전된 화상

애플에서의 고해상도 그래픽 화상은 어떤 한 컴퓨터 메모리 블록에 저장되어 있는 숫자들의 비트 패턴(bit pattern)을 그대로 디스플레이한 것이다. 이 때 화상은 1의 값을 갖는 비트들에 해당하는 화소들을 ON시키고 0의 값을 갖는 비트들에 해당하는 화소들은 OFF시킴으로써 구성된다.

만약 고해상도 그래픽 메모리의 내용들을 일괄적으로 바꾸어 주면, 재미있는 화상으로 바뀌게 될 것이다. 반전된 화상은 ON상태의 모든 화소들을 OFF 상태로 바꾸고, 또 OFF상태의 모든 화소들은 ON상태로 바꾸어 줌으로써 얻을수 있다. 이렇게 하기 위해서는 컴퓨터 메모리의 각 비트들을 0은 1로, 1은 0으로 모두 바꾸어 주어야 한다.

이와 같은 것을 페이지 1에서 할려면 다음과 같이 한다.

```
FOR I = 8192 TO 16383: POKE I, 255 - PEEK(I): NEXT I
```

또, 페이지 2에 대해서는 다음과 같다.

```
FOR I = 16384 TO 24575: POKE I, 255 - PEEK(I): NEXT I
```

대화형 프로그램에서는 INPUT명령문이나 GET명령문을 이용하여 사용자로부터 데이타를 입력한다.

만약 이러한 문장이 수행되면, 컴퓨터는 입력을 받을 때까지 기다렸다가 그 다음문장들을 수행한다. 가끔 프로그램 수행중에 특별한 키, 예로서 [ESC] 키가 눌려지면 수행을 끝나는 경우가 있다.

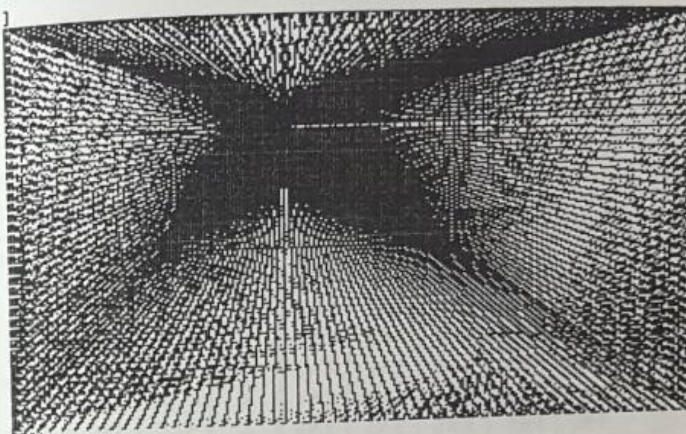
다음의 프로그램이 이런 경우이다.

```
* PROGRAM 2.38

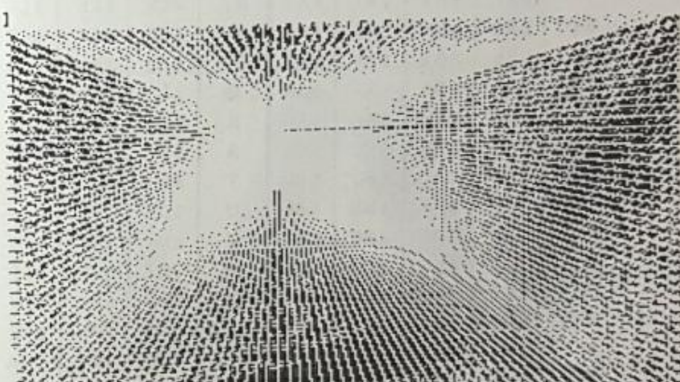
 1 REM ***********************
 2 REM *    M O I R E    *
 3 REM ***********************
 4 REM
 10 HGR2 :A = 5
 20 X1 = RND (1) * 279:Y1 = RND
     (1) * 191
 30 C = C + 1: IF C = 8 THEN C = 0

 40 HCOLOR= C:Y = 191
 50 FOR X = 0 TO 279 STEP A
 60 HPLOT X,191 TO X1,Y1 TO 279 - X,0
 70 NEXT X
 80 FOR Y = 0 TO 191 STEP A
 90 HPLOT 279,191 - Y TO X1,Y1 TO 0,Y
 100 NEXT Y
 110 FOR I = 1 TO 500: NEXT I
 120 IF PEEK ( - 16384) = 155 THEN
     POKE - 16368,0: TEXT : HOME : END
 130 GOTO 20
```

〈원래의 그림〉



〈반전된 그림〉



여기서 사용자로 부터 입력을 받아들이기 위해 프로그램의 수행을 일단 정지 시키지 않고, 프로그램이 일정하게 규칙적으로 메모리번지 -16384의 내용을 PEEK(-16384)에 의해 점검하도록 되어 있다. 이 메모리번지에는 눌러진 키의 값이 저장되게 되어 있다. 만약 ESC 키가 눌러졌다면 PEEK(-16384) = 155에 의해 프로그램은 수행을 끝내게 된다 (행번호 120). 그렇지 않으면 수행을 계속한다. 다음의 ASCII코드에 대한 표를 참조하라.

행번호 120에서는 주의할 점이 한 가지 더 있다.

어떤 키가 눌려지면, 그것에 해당하는 코드가 메모리번지 -16384에 기억되어 남아 있게 된다. 키의 값이 읽혀질 때,

다음 키의 값을 받기 위해

```
POKE - 16368,0
```

명령으로 키보드의 스트로우브(strobe)를 소거(clear)한다.

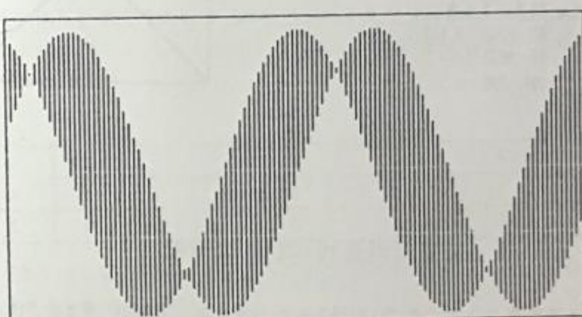
표2-8 키와 이와 관련된 아스키 코드

Key	Alone	CTRL	SHIFT	Both	Key	Alone	CTRL	SHIFT	Both
space	$A0	$A0	$A0	$A0	RETURN	$8D	$8D	$8D	$8D
0	$B0	$B0	$B0	$B0	G	$C7	$87	$C7	$87
1 !	$B1	$B1	$A1	$A1	H	$C8	$88	$C8	$88
2 "	$B2	$B2	$A2	$A2	I	$C9	$89	$C9	$89
3 #	$B3	$B3	$A3	$A3	J	$CA	$8A	$CA	$8A
4 $	$B4	$B4	$A4	$A4	K	$CB	$8B	$CB	$8B
5 %	$B5	$B5	$A5	$A5	L	$CC	$8C	$CC	$8C
6 &	$B6	$B6	$A6	$A6	M	$CD	$8D	$DD	$9D
7 '	$B7	$B7	$A7	$A7	N ∧	$CE	$8E	$DE	$9E
8 (	$B8	$B8	$A8	$A8	O	$CF	$8F	$CF	$8F
9)	$B9	$B9	$A9	$A9	P@	$D0	$80	$C0	$80
: *	$BA	$BA	$AA	$AA	Q	$D1	$91	$D1	$91
; +	$BB	$BB	$AB	$AB	R	$D2	$92	$D2	$92
, <	$AC	$AC	$BC	$BC	S	$D3	$93	$D3	$93
− =	$AD	$AD	$BD	$BD	T	$D4	$94	$D4	$94
. >	$AE	$AE	$BE	$BE	U	$D5	$95	$D5	$95
/ ?	$AF	$AF	$BF	$BF	V	$D6	$96	$D6	$96
A	$C1	$81	$C1	$C1	W	$D7	$97	$D7	$97
B	$C2	$82	$C2	$C2	X	$D8	$98	$D8	$98
C	$C3	$83	$C3	$C3	Y	$D9	$99	$D9	$99
D	$C4	$84	$C4	$C4	Z	$DA	$9A	$DA	$9A
E	$C5	$85	$C5	$C5	→	$88	$88	$88	$98
F	$C6	$86	$C6	$C6	←	$95	$95	$95	$95
					ESC	$9B	$9B	$9B	$9B

문제 129. 화면에 SIN그래프와 COS그래프를 90°정도의 위상차를 갖도록 하여 겹쳐 그린후, 두 곡선에 의해 이루어지는 폐곡선의 안쪽을 청색으로 칠하는 프로그램을 작성하라.

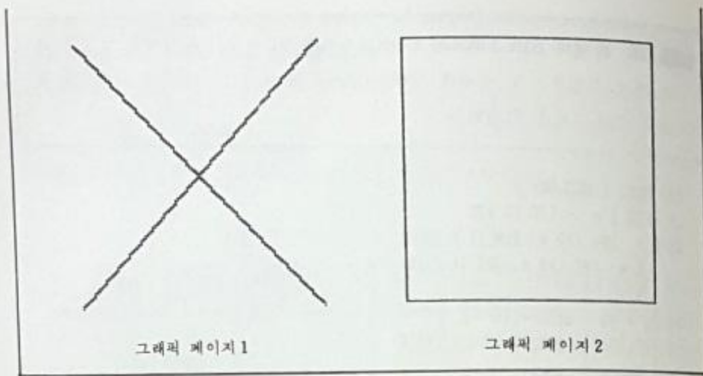
```
10  HGR2 : HCOLOR= 2
20  FOR I =  - 120 TO 120
30 S =  INT (70 *  SIN (I / 20)):
    C =  INT (70 *  COS (I / 20)
    )
40 SS = 96 - S:SC = 96 - C
50  HPLOT I + 140,SS TO I + 140,S
    C
60  NEXT I
70  HCOLOR= 3
80  HPLOT 20,20 TO 260,20 TO 260,
     170 TO 20,170 TO 20,20
90  END

]
```

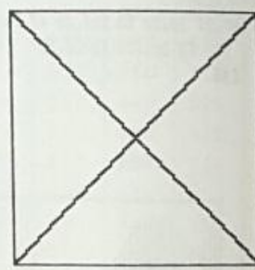


문제 130. 애플에는 두개의 고해상도 그래픽 페이지가 마련되어 있어, HGR 방식에서는 그래픽 페이지 1에 화상이 그려지고, HGR 2 방식에서는 페이지 2에 화상이 그려진다. 소프트 스위치를 이용하여 화면에 디스플레이 될 페이지를 선택할 수 있다.

다음과 같이 그래픽 페이지 1에는 X모양, 그래픽 페이지 2에는 정사각형을 그린후, 이 두 그래픽 페이지가 번갈아 가면서 디스플레이되는 프로그램을 작성하라.




```
10  HGR : HCOLOR= 3
20  HPLOT 90,40 TO 190,40 TO 190,
      160 TO 90,160 TO 90,40
30  HGR2
40  HPLOT 90,40 TO 190,160: HPLOT
      190,40 TO 90,160
50  FOR I = 1 TO 100
60  FOR J = 1 TO 50: NEXT J
70 A = 1 - A
80  POKE  - 16299 - A,0
90  NEXT I
100  END
```



2-6 도형정의표에 의한 그래픽

도형정의표 그래픽 화상을 만드는 한 방법으로는 애플소프트 BASIC 명령인 HPLOT과 HPLOT TO를 사용하는 것이다. 그러나 자주 사용되는 그래픽 요소들을 디스플레이 하는데는 도형의 데이타인 **圖形定義表** (shape table)을 이용하는 것은 유용하다.

도형정의표는 점의 이동방향을 나타내는 정보,즉 벡터테이블(vector table) 의 형태로 정의하며 테이블의 수를 증가시키면 여러개의 도형을 만들 수도 있다. 그리고 도형의 확대, 회전등도 모두 실행시킬 수 있다.

도형(shape)에 대한 정의는 메모리에 있는 연속된 바이트들에 저장되어 있는 점 벡터들의 계열로 구성한다. 즉 각 도형들은 그 도형을 그리는데 쓰이는 벡터들을 나타내는 코드들의 조합으로 구성되는데, 이들 각각의 벡터들은 점을 단위거리만큼 상하좌우로 이동시키게 되는 것으로, 이때 점을 찍고 이동하는 경우와 점을 찍지않고 이동하는 경우로 나눌 수 있다.

다시 말하면 도형 정의의 구조인 각 바이트의 구성은 3개 부분으로 되어 있으며 각 부분이 「점벡터」를 정의한다. 정의된 내용은 점의 유·무 및 상·하·좌·우의 동작방향이다.

표2-9에 도형을 정의하는데 사용될 수 있는 8개의 벡터들이 그에 대한 코드와 함께 나타나 있다.

표 2-9 도형정의 벡터의 종류

벡 터	내 용	코 드	
		2진	10진
↑	점을 찍지 않고 위로 이동	000	0
→	점을 찍지 않고 오른쪽으로 이동	001	1
↓	점을 찍지 않고 아래로 이동	010	2
←	점을 찍지 않고 왼쪽으로 이동	011	3
↑ (점)	점을 찍고 위로 이동	100	4
→ (점)	점을 찍고 오른쪽으로 이동	101	5
↓ (점)	점을 찍고 아래로 이동	110	6
← (점)	점을 찍고 왼쪽으로 이동	111	7

이 8가지의 코드는 최종적으로 8비트의 데이타로서 할당되는데, 하나의 코드는 3비트씩 요하게 되므로 1바이트 중에 3비트의 코드를 2개 넣으면 2비트가 남게 되므로 비경제적이다.

section	C		B			A		
bit code	7	6	5	4	3	2	1	0
기 호	D	D	P	D	D	P	D	D

※D는 이동의 방향을 나타내는 비트
P는 점의 유무를 나타낸다.

그림2-21

따라서 section C의 최상의 비트, 즉 존재치 않는 비트 코드「8」를 항상 0로 하여 점을 찍지 않고 이동하는 것으로 하고 남은 2비트를 이동 방향의 지정에 사용하기로 하면 8비트 전체를 활용할 수 있게 된다. 그러므로 이것은 꼭 이 부분에 와서 도시하지 않고 이동하라는 순번이 되었을 때만 이용된다. 또 다음에 기술하는 이유 때문에 점을 찍지 않고 위로 이동하는「00」은 정의할 수도 없다.

각 바이트는 최아래 비트에서 최상의 비트로 향하여, 즉 section A에서 section C를 향하여 처리된다. 이 때 어느 바이트중에 남은 section이 전부 0이면 점을 찍지 않고 위로 이동된다고 볼 수 없으므로 무시된다. 즉 section C으로는

	secC	secB	sec A	
△	00	000˙	000	도형정의표의 끝에서만 사용.

×	00	000	101	sec B가 000일때 sec C는 00으로 안된다.
○	10	000	101	
○	10	000	101	
○	11	000	101	

×	00	000	000	sec A, B가 000인 것은 sec C가 00이 아닌 경우만 더 이상 점을 찍지 않고 위쪽 이외의 방향으로 이동할 때만 성립된다.
○	01	000	000	
○	10	000	000	
○	11	000	000	

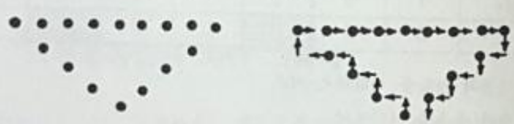
그림 2-22 사용할 수 없는 코드의 조합

「00」이 정의되지 않으므로 section C가 좌우 또는 아래로 이동된다고 정의되어 있지 않는 한 section B는.「000」으로 정의할 수 없다.

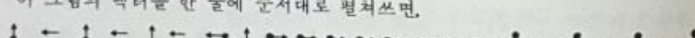
같은 방법으로 section B와 C가 전부 「0」일때는 section A도 「000」을 정의할 수 없다. 이것은 1바이트전부 가 「0」인 경우는 도형정의표의 끝을 의미하기 때문이다.

그렇기 때문에 점을 찍지 않고 연속하여 위로 이동하는 것은 section A와 B에서 최고 2회까지만 허용되며,이렇게 되고 난 다음에는 반드시 점을 찍지 않고 위쪽 이외의 다른 방향으로 이동해야 한다.

자, 다음의 3각형 모양을 통하여 도형정의에 대해 알아보자. 이것을 알기쉽게 모눈종이에 옮기고 그 도형정의 벡터로 표시해 보자.



이 그림의 벡터를 한 줄에 순서대로 펼쳐쓰면,



으로 되며, 이것을 section A, B, C로 할당하고, 이것을 코드화하여 16진수로하면 그림2-23과 같다.

그림2-23의 도형정의 벡터의 코드화는 도형정의표로서는 불완전한 것이다. 그 이유는 도형정의표는 도형을 여러가지로 정의할 수 있게 하기 위해서 각 도형표의 시작번지를 지정하는 색인(index)을 붙여야 하는데, 붙이지 않았기 때문이다.

기호표시 section C	기호표시 section B	기호표시 section A	2진수 코드 section C	2진수 코드 section B	2진수 코드 section A	16진수코드
	←	•↑		011	100	1C
	←	•↑		011	100	1C
	←	•↑		011	100	1C
		←•			111	07
	•→	↑		101	000	28
	•→	•→		101	101	2D
	•→	•→		101	101	2D
	•→	•→		101	101	2D
	•↓	•→		110	101	35
	•↓	←		110	011	33
	•↓	←		110	011	33
	•↓	←		110	011	33
			00	000	000	00

← 도형정의의 끝을 나타냄.

그림 2-23 도형정의 벡터의 코드화

실제로 도형정의표는 그림 2-24와 같이 구성되며 시작번지에 대한 상대번지를 색인으로 갖추고 있다.

도형표의 시작(start)번지를 S라고 한다면 S번지들의 내용들은 도형정의에 대한 색인으로서, 시작번지에 얼마만큼 떨어진 메모리에 저장되어 있는가 하는 간격을 나타낸다.

여기서 도형정의는 하나이므로 시작번지 S에는 도형정의의 갯수인 1이 $(01)_{16}$으로 저장되며, 이도형정의는 S+4부터 도형정의 1번이 시작되므로 D1값은 4가 된다. 그러므로 색인 바이트 S+2는 S+3번지와 함께 $(0004)_{16}$를 저장한다.

이 결과를 그림2-25에 나타내었다.

도형정의표는 어느정도 임의의 메모리 번지에 저장할 수 있는데, 이 때에는 반드시 표의 시작번지를 도형을 그리는 명령어 DRAW, XDRAW에게 알려 주어야 한다. 그래서 표의 시작번지는 메모리 번지 $00E8(232)와 $00E9(233)에 기억되어야 한다. 이 메모리 번지에 올바르게 저장되었는가를 프로그래머는 항상 확인하여야 한다.

예를 들어 시작번지를 $1ABC부터 두었다고 하면 모니터에서는

```
*E8: BC 1A
```

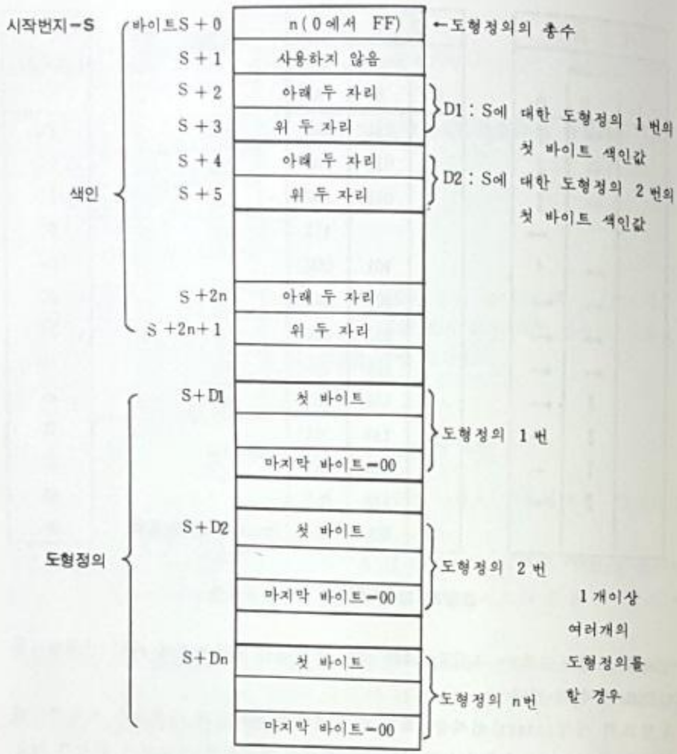


그림 2－24 도형정의표(shape table)의 메모리 저장 형태

가 되고 애플소프트 BASIC에서는

```
]POKE 232,188:POKE 233,26
       ↑            ↑
   $ BC＝188     $ 1A－26
```

로 설정할 수 있다.

일반적으로 번지 A로부터 기억되어 있는 도형정의표에 대하여, 메모리 번지 $ 00E8(232)에는 A－256＊(INT(A/256))이 저장되고, 메모리 번지 $ 00E9(233)에는 INT(A/256)이 저장되어야 한다. 이 값들은 표가 디스크로 부터 적재될 때 지정될 것이다. 그런데 만약 메모리 번지 232와 233의 내용은 적절한 때에 적절하게 바꾸어 주기만 하면 동시에 여러 개의 도형정의표를 메모리상에 적재시켜 놓고 사용할 수도 있다.

번지	값	
$1ABC	01	←도형의 갯수. 여기서는 1개
$1ABD	00	
$1ABE	04	시작번지에 대한 도형정의 1번의 색인값
$1ABF	00	
$1AC0	1C	←첫 바이트
1	1C	
2	1C	
3	07	
4	28	
5	2D	
6	2D	도형 ▽의 도형정의 1번
7	2D	
8	35	
9	33	
A	33	
B	33	
C	00	←마지막 바이트

그림 2-25 도형정의표의 실제 예

도형정의표의 시작번지가 정의되면 애플소프트의 명령어인 DRAW와 XDRAW를 사용할 수 있는데, 이런 명령을 사용하려면 미리 다음과 같은 값을 지정해 두어야 한다.

1. HCOLOR

도형들이 회전하거나 크기 조절이 되면 약간씩 이그러지는 것을 알 수 있다. 그리고 대부분의 도형들은 수직선을 포함하게 되므로 컬러를 사용하는데 약간의 곤란한 문제가 생긴다. 그러므로 도형을 디스플레이 할 때는 흰색을 사용하는 것이 좋다.

2. ROT=n

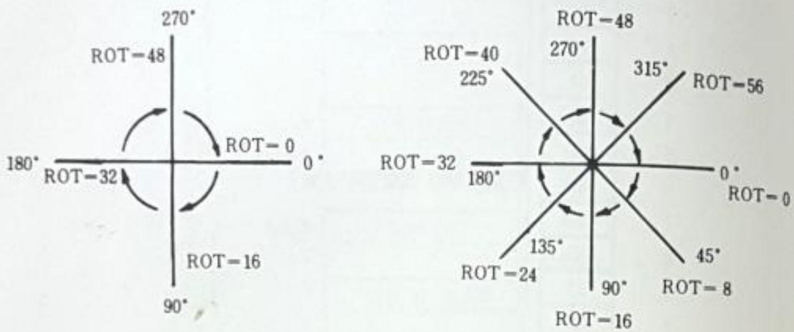
DRAW나 XDRAW에 의해서 그려지는 도형의 회전각을 지정한다. n은 0에서 255까지 사용할 수 있으나 실제의 지정범위는 0에서 65까지이며, 나머지 값일때

표 2-10 SCALE에 따른 ROT값과 회전각

SCALE=1	ROT=	0		16		32		48	
	회전각	0°		90°		180°		270°	
SCALE=2	ROT=	0	8	16	24	32	40	48	56
	회전각	0°	45°	90°	135°	180°	225°	270°	315°

는 같은 것이 되풀이 된다. 실제로 64의 값에 대응하는 모든 방향으로 그려지는 것이 아니고 표2-10과 같이 설정된다.

즉 이 범위에 속하지 않는 SCALE=1에서 ROT=18인 경우는 지정치와 가장 가까운 값인 16으로 설정된다. 기본적으로 R=0는 원위치, R=16은 90°회전, R=32는 180°회전을 나타낸다. 이 때 회전방향은 시계방향이다.



(a) SCALE=1 일때 (b) SCALE=2 일때

그림 2-26 ROT의 값과 회전각도

SCALE=

3. SCALE=n

DRAW나 XDRAW로 그려진 도형의 크기를 0부터 255까지로 정한다. n은 원래 shape definition의 n배 크기를 나타낸다. n=0가 가장 큰 배율이 되나 이 경우 정의된 도형의 크기가 1이므로 축소는 되지 않는다.

DRAW

4. DRAW n AT X, Y

좌표(X, Y)에 도형정의를 그린다. 다시 말하면 SHLOAD등으로 카세트 테이프로 부터 읽어 드리거나 도형정의표가 메모리내에 정의되어 있는 도형을 PLOT하기 위한 명령으로 여기서 만일 AT X, Y가 생략되면 바로 전의 위치에 그리게 되며 n의 범위는 0에서 255까지이다.

여기서 1~3이 설정되어 있지 않으면 DRAW는 실행되지 않는다. 실제의 프로그래밍에서 1~3을 흔히 잊어 버리기 쉬우므로 주의해야 한다.

이렇게 하여 그려진 도형을 지우기 위해서는 XDRAW를 사용하는데,

XDRAW

5. XDRAW n AT X, Y

DRAW에서 그려진 색의 보색(반대색)을 그리므로 그림이 지워지는 것처럼 보인다.

XDRAW는 PLOT되는 색이외에는 DRAW와 똑 같다. PLOT하는 좌표의 현재색과 보색관계가 되어

흑색 ↔ 백색

자색 ↔ 녹색

등색 ↔ 청색

과 같이 합쳐지면, 배경색이 있는 경우 XDRAW를 두번 되풀이 하므로서 도형정의를 그리거나 지울 수 있어서 매우 편리하다.

```
* PROGRAM 2.39
 10 HGR : HCOLOR= 3
 20 FOR R = 1 TO 50
 30 ROT= R
 40 SCALE= R
 50 DRAW 1 AT 139,79
 60 NEXT R
```



여기서 쓰인 도형정의표는 위에서 정의된 것이며, 번호는 1이다.

SHLOAD

DRAW나 XDRAW는 도형정의표가 있어야만 도형을 그리므로 도형을 데이타에 미리 넣어두어야 한다. 일단 만든 표(table)을 카세트 테이프에 기록하여 놓았을 때는 SHLOAD 명령으로 읽어들일 수 있다.

이때, 표의 데이타는 HIMEM: 으로서 지정된 번지로 부터 아래를 향해서 저장되며 읽어들이기가 끝나면 HIMEM:은 도형정의표 아래에 다시 세트되고 BASIC 프로그램이 도형정의표를 파괴하지 않도록 보호한다. DRAW 등으로 도형을 그릴 때는 표의 시작번지를 지정해 두어야 하지만 이것은 표를 읽어들인 뒤에 자동으로 세트된다.

도형정의표를 이용하여 사용자가 키보드 에 의한 방법으로 컴퓨터에게 원하는 그림을 그리게 하는 프로그램의 예를 보자.

화면의 어떤 부분에 그림을 그리기 위해서는 회전하는 하나의 선분인 커서가 사용되는데, 커서는 Ⓒ, Ⓓ, Ⓔ 키에 의해 세가지 모드중의 하나로 동작한다. 즉 C 모드에서는 커서만 회전할 뿐 그림은 그려지지 않는다. 만약 D (draw, 그리기) 모드가 선택되면, 커서가 지나가는 부분에 점정색의 선분들이 그려짐으로서

화면의 그 부분을 지우는 효과를 나타낸다. 왜냐하면 화면의 바탕색이 검정색이기 때문이다. D모드에서 사용될 컬러는 키 1, 2, 3, 4, 5, 6에 의해 결정된다.

커서의 크기는 L (lengthen) 키에 의해 길어지기도 하고 S (shorten) 키에 의해 줄어들기도 한다. 길이를 빨리 변형시키려면 L 키나 S 키와 함께 REPT 키를 누르면 된다.

커서가 회전하는데 있어서의 기준점은 I, J, K, M 키에 의해 그 위치가 바뀌어 진다. 이 키들의 역할은 다음과 같다.

I : 위로 이동

J : 왼쪽으로 이동

K : 오른쪽으로 이동

M : 아래로 이동

앞에서와 마찬가지로 위치를 빨리 이동시키려면 REPT 키와 함께 사용하면 된다.

이 프로그램에서 간단한 도형정의표가 행번호 10부터 40까지에서 주어지는데 정의된 도형은 회전하는 커서이다. 이 도형의 크기, 회전, 색, 위치등은 프로그램 내에서 변경되어 사용된다.

프로그램을 완전히 타이핑한 후에 RUN을 치고 RETURN 키를 누르면 화면은 흑색으로 되고 중앙 근처에는 깜박이면서 회전하는 막대가 나타날 것이다.

```
* PROGRAM  2.40
 1  REM  ********************
 2  REM  *                  *
 3  REM  * ARTIST'S AID     *
 6  REM  *                  *
 7  REM  ********************
 10  POKE 768,1: POKE 770,4
 20  POKE 771,0: POKE 772,4
 30  POKE 773,0
 40  POKE 232,0: POKE 233,3
 50  HGR : POKE  - 16302,0:C = 3:S
      = 25:X = 140:Y = 96
 60  FOR I = 1 TO 64
 70 A =  PEEK ( - 16384): POKE  -
     16368,0
 80  IF A = 155 THEN  TEXT : HOME
     : END
 90  IF A = 177 THEN C = 1
 100  IF A = 178 THEN C = 2
 110  IF A = 179 THEN C = 3
 120  IF A = 180 THEN C = 4
 130  IF A = 181 THEN C = 5
 140  IF A = 182 THEN C = 6
 150  IF A = 202 THEN X = X - 1: IF
      X < 0 THEN X = 279
 160  IF A = 203 THEN X = X + 1: IF
      X > 279 THEN X = 0
 170  IF A = 201 THEN Y = Y - 1: IF
      Y < 0 THEN Y = 191
 180  IF A = 205 THEN Y = Y + 1: IF
      Y > 191 THEN Y = 0
 190  IF A = 211 THEN S = S - 1: IF
      S < 1 THEN S = 1
 200  IF A = 204 THEN S = S + 1
 210  IF A = 195 THEN F = 0
 220  IF A = 196 THEN F = 1
 230  IF A = 197 THEN F = 2
 240  IF A = 200 THEN  HGR2
 250  HCOLOR= C: ROT= I: SCALE= S
 260  IF F = 0 THEN  HCOLOR= 3: XDRAW
      1 AT X,Y: XDRAW 1 AT X,Y: GOTO
      300
 270  DRAW 1 AT X,Y
 280  IF F = 1 THEN 300
 290  HCOLOR= 4: DRAW 1 AT X,Y
 300  NEXT I
 310  GOTO 60
```

이 인공의 붓을 사용하기 위해 다음과 같이 해 보자.

- 그림을 그리기 위해 D 키를 눌러 보아라.
- 약 2초동안 K 키와 REPT 키를 동시에 눌러 보아라. 그러면 커서가 오른쪽으로 이동하며 그림을 그릴 것이다.
- 색을 바꾸기 위해 4 키를 눌러라.
- M 키와 REPT 키를 눌러 아래쪽으로 움직이며 그림을 그려라.
- 그림의 그리기를 중지하기 위해서는 C 키를 눌러라. 그러면 커서는 보이지만 그것이 회전하는 자취에 따라 그림은 그려지지 않는다.
- J 키와 REPT 키를 동시에 눌러 왼쪽으로 이동하게 하라.
- S 키와 REPT 키를 동시에 눌러 커서의 길이를 줄인다.
- 이제 그림 그리기를 다시 시작하기 위해 D 키를 누른다.

이제부터는 당신의 상상력과 창조력에 따라 키들을 사용해 보라. 그림과 같이 여러분의 의도에 따라 임의의 그림을 그릴 수 있을 것이다. 만약 이 프로그램을 중지시키고 싶을 경우에는 ESC 키를 눌러라. 그러면 그림은 사라지고 이 프로그램 수행전에 주었던 명령이나 프로그램의 일부가 들어있는 텍스트페이지가 화면에 나타나고 수행이 중지될 것이다.

만약 당신이 그림을 다시 보기위해 그래픽페이지로 돌아가려고 할 때, HGR을 치면 안된다. 왜냐하면 HGR은 화면을 고해상도 그래픽 페이지로 돌아가도록 하지만, 그 곳의 내용을 지워버리게 된다.

이러한 것은 다음과 같은 명령들은 사용하여 해결할 수 있다.

```
]POKE -16297,0:POKE -16302,0:POKE -16304,0:POKE -16300,0
```

만약 화상을 나중에 디스플레이시켜 보기위해 디스크에 기억시켜 놓고 싶을 경

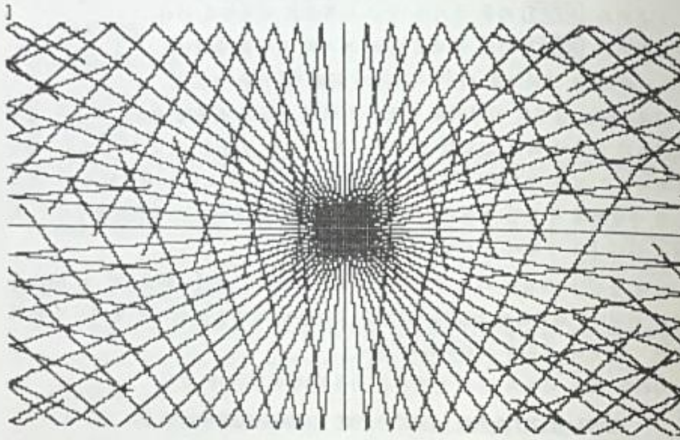


우에는 먼저 그 그림의 이름(NAME)을 결정하고 다음과 같이 타이프 한다.

```
]BSAVE NAME, A$2000, L$2000
```

그리고 후에 그 그림을 다시 보고 싶은 경우에는 다음과 같은 명령들을 사용한다.

```
HGR:POKE -16302,0:PRINT CHR$( 4):"BLOAD NAME, A$2000"
```



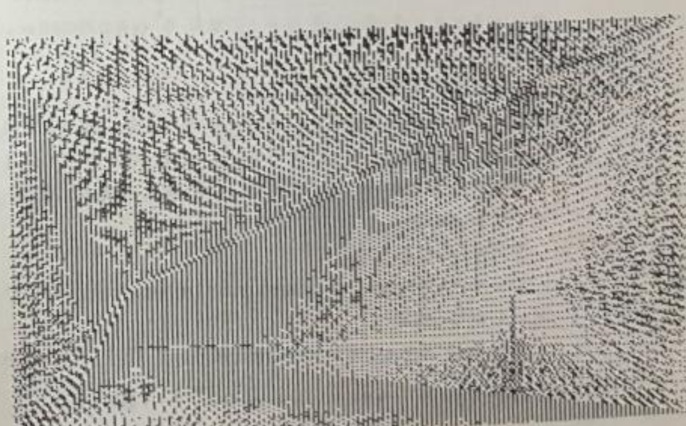
문제 131. 고해상도 그래픽 모드를 이용하여 회전각은 1에서 64, 스케일은 1으로하고 번호는 1, 좌표는 (139, 79)을 중점으로 한, 다음 도형과 같이 출력되도록 프로그램을 작성하라.



```
10  REM  DRAW N AT X,Y
20  HGR
30  HCOLOR= 3
40  FOR R = 1 TO 64
50  ROT= R
60  SCALE= 1
70  DRAW 1 AT 139,79
80  NEXT R
```

3 apple

특수기능



3-1 기계어의 기초

PEEK와 POKE PEEK(번지)는 애플의 기억장치의 내용을 꺼내보는데 사용된다. 즉 ()내에 10진수로 지정된 메모리 번지의 내용(00~FF)을 10진수(0~255)로 빼낸다.

POKE 번지, 수치는 PEEK와 반대로「번지」로 지정된 번지에「수치」를 써 넣을 수 있다. 양쪽 다 10진수를 사용하지만「번지」는 애플소프트에서는 -65535에서 65535까지의 범위밖에 사용할 수 없다. 그러나 메모리 장치 중에서 사용자의 프로그램이 기억되는 곳이 RAM이므로 48K시스템은 0에서 49152번지 까지를 의미한다. 그렇지만 소프트 스위치등의 특정한 번지의 액세스(access)로 사용하는 경우에는 이 범위를 벗어난다.

```
* PROGRAM 3.1

 10 PRINT PEEK (2000)
 20 POKE 2000,65
 30 PRINT PEEK (2000)
 40 END

 ]RUN
 160
 65
```

여기서 행번호 10은 10진수로 2000번지에 기억되어 있는 값을 프린트하라는 문으로 이 번지의 내용은 10진수로 160이다. 그러나, 행번호 20에서 2000번지에 65를 새로 POKE 명령으로 써 넣었으므로 행번호 30에서 다시 PEEK 명령으로 2000번지의 내용을 프린트하면 바뀐 데이타 65가 출력된다.

USR 그리고 기계어 루틴으로 갈때 BASIC 프로그램중의 수치를 기계어 루틴으로 보내서 사용하는 수가 있다. 이때 사용하는 명령어가 USR이고 ()속에 지정된수치는 애플소프트가 연산등에 부동소숫점의 주 어큐므레이터(accumulator)에 저장된다.

이 동작을 순서적으로 나열하면,

① 수치를 보통 FORMAT로 변환시켜 부동소숫점 어큐므레이터에 저장시킨다.

② $ 0A(10) 번지에서 기계어루틴으로 뛴다. 단지 3바이트의 기계어와 몇번지로 분기한다는 **JMP** 명령을 넣을 수 있다. 예로써 $4C, $00, $03 등이다(16진수로 4C 00 03).

③ 기계어루틴을 실행한다.

④ 기계어루틴의 끝에 있는 **RTS**(RETURN을 뜻하는 기계어) 명령으로 애플소프트로 되돌아간다.

⑤ 이 때에 부동소숫점의 주 어큐므레이터의 값이 USR(수치)의 값으로 된다.

```
* PROGRAM  3.2
10  POKE 10,76: POKE 11,0: POKE 1
      2,3
20  POKE 768,32: POKE 769,141: POKE
      770,238: POKE 771,96
30  INPUT "DATA? ";X
40  PRINT "SQR(DATA)= "; USR (X)
50  END

]RUN
DATA? 121
SQR(DATA)= 11
```

이 프로그램 중에서 X=121은 SQR함수가 기계어로 되어 있을 때 SQR (121)과 같이 쓸 때의 변수값이며, 행번호 40에서 USR 명령을 만나게 되면 메모리 10, 11, 12번지를 거쳐 768번지로 가서 서브루틴인 사용자 프로그램, 예로 SQR을 수행한 후에 행번호 50으로 되돌아가게 된다.

위의 프로그램 예에서 행번호 10~20을 어셈블러(assembler)로 쓰면 다음과 같다.

```
000A-   4C 00 03    JMP   $0300
0300-   20 8D EE    JSR   $EE8D
0303-   60          RTS
```

라는 기계어 프로그램을 메모리에 set시켜 수치데이타 X를 보내는 것이다. 이 프로그램을 그림으로 나타내면 다음과 같다.

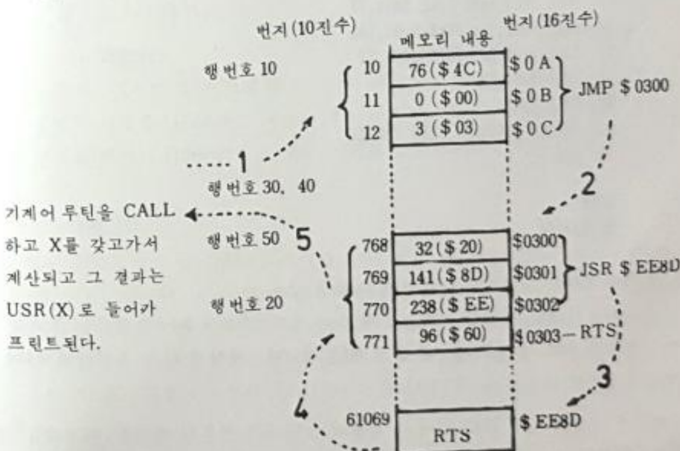


그림 3-1 예를 든 프로그램동작과 USR(X)의 동작

여기서 $EE8D부터는 애플소프트 ROM내의 루틴으로 부동소숫점의 주 어큐므레이터의 값을 평방근으로 계산한 후, 다시 소숫점의 주 어큐므레이터로 보내는 동작을 한다.

CALL

그러면 기계어 서브루틴을 실행시키는 역할을 하는 CALL에 대해서도 알아 보자. 이것은 BASIC으로 프로그램을 짜기 쉬우나 실행속도가 느리기 때문에 고속처리가 필요한 경우에는 기계어로 처리 프로그램을 작성하여 적당한 번지에 넣어 두었다가 BASIC 프로그램에서 CALL로 선두 번지를 부르도록 하므로써 BASIC과 기계어의 특징을 살릴 수 있다. 기계어 루틴에서 BASIC 프로그램으로 돌아올 때는 기계어 루틴끝에 RTS(return subroutine)인 기계어 명령을 넣어 두어야 한다.

이렇게 하면 모든 번지($0000~$FFFF)를 지정할 수 있을 뿐만 아니라, 모니터나 ROM속의 루틴도 이용할 수 있다. 그래픽이나 게임등에서는 기계어 프로그램의 도움을 받는 경우가 많다. 이런 기계어 루틴은 모니터의 메모리에 기억능력을 이용해서 기억시킬 수도 있으나 DATA, READ, POKE를 사용해서 BASIC으로 기억시키고 CALL로서 실행하는 방법이 많이 이용된다.

CALL과 USR의 차이는 CALL은 매개변수 값을 기계어 서브루틴에 전달할 수 없고, USR은 기계어 서브루틴을 실행시키는데 시작번지 정의 과정이 좀 복잡한 반면에 CALL은 바로 시작번지를 정의할 수 있다.

기계어 서브루틴은 다음의 프로그램 예처럼 POKE문을 이용하여 필요한 기계어를 메모리에 기억시킴으로써 정의된다.

```
* PROGRAM 3.3
 10 POKE 5000,169: POKE 5001,77
 20 POKE 5002,10: POKE 5003,141
 30 POKE 5004,00: POKE 5005,32
 40 POKE 5006,96
 50 CALL 5000
 60 PRINT "8192>>"; PEEK (8192)
 70 END

 ]RUN
 8192>>154
```

이 프로그램을 그림으로 나타내면 그림 3-2와 같다.

여기서 누산기의 내용을 왼쪽으로 한자리 옮기면(shift left) 2배하는 효과를 얻기 때문에 ASL 명령을 사용해서 2배한 것이다. 따라서 77을 2배한 결과 154가 8192번지에 있음을 알 수 있다.

HIMEM:

메모리 조작에 있어 BASIC에서 사용되는 최대의 메모리 번지를 지정하는 명령을 HIMEM:이라하며 이 명령은 BASIC 프로그래밍을 할 때, 사용자가 사용할 수 있는 메모리의 영역을 제한하기 위해서 사용하는 것으로 사용자가 BASIC으

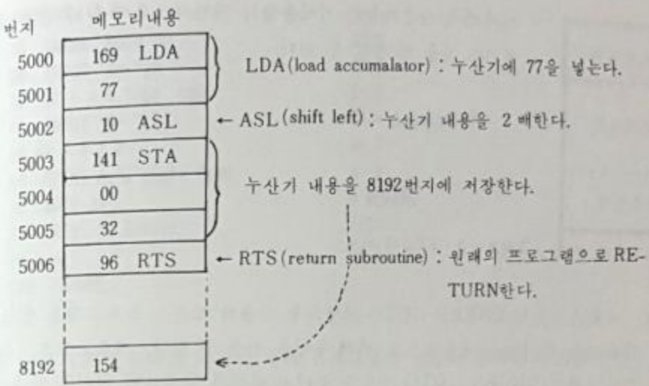


그림 3-2

로 사용할 수 있는 최상의 번지를 지정하여, 그 번지보다 큰 번지는 사용자가 쓸 수 없게 하는 시스템 명령이다.

```
]HIMEM:65532
```

이 명령은 사용자가 BASIC으로 65532번지 이상은 사용 못하게 한다. 일반적으로 이 HIMEM:명령은 기계어로 작성된 프로그램과 BASIC 프로그램이 서로 상대방을 파괴하지 않도록 방지하는데 적절히 사용된다.

LOMEM:

HIMEM: 과는 반대로 BASIC 프로그램에서 사용할 수 있는 최소번지를 지정하는 명령은 LOMEM:이다.

```
]LOMEM:181
```

FRE

HIMEM: 에서 기술한 것과 마찬가지로 애플소프트에서는 사용자가 사용 가능한 영역의 한계를 HIMEM을 지정해 줌으로서 결정되며, 프로그램중에 문자변수가 정의될 때마다 HIMEM으로 부터 아래로 순차적으로 그들의 영역을 확보해 간다.

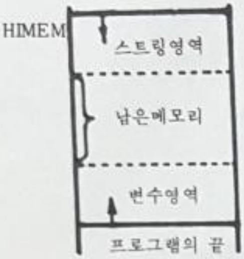
```
FRE (0)
```

일반 프로그램이 많아지면 프로그램 끝이 변수 및 배열의 포인터 영역까지 위를 향해서 늘어난다. 이 사이가 당신의 사용가능한 기억용량이 되는 것이다. FRE(X)는 사용가능한 기억용량을 나타내며 ()내는 어떤 숫자를 사용해도 상관없으나 다른것에 지장이 없도록 FRE(0)라고 하는 것이 좋다.

BASIC 프로그램이 기계어 루틴이나 그래픽의 메모리 영역을 범하지 않게 위해서

```
X = FRE (0)
```

를 실행한다. 이로서 과거에는 사용했으나 지금은 사용하지 않는 낡은 문자가 없



어지면서 사용가능한 기억용량이 많아지므로서 위와 같은 문제가 생기는 것을 방지할 수 있다.

그림 3 - 3 FRE의 의미

ONERR GOTO 애플소프트의 ONERR GOTO 코맨드를 사용해 보통은 프로그램을 중단시키는 메세지를 처리하는 애플소프트 에러 루틴을 만들 수 있다. 애플소프트 BASIC 프로그램의 ONERR GOTO 명령의 뒤에서 에러가 발생하면 에러의 종류에 대응한 코드번호가 10진 메모리번지 222에 기억된다. 이것은 애플소프트가 애플소프트 에러 메세지에 대응하는 코드를 기억하는 것과 같은 메모리번지이다.

```
Y = PEEK (222)
```

라는 코맨드는 애플소프트의 ONERR GOTO 점프를 시키는 에러에 대응하는 ONERR GOTO 코드를 Y에 넣는다.

에러가 생기거나 혹은 에러가 발생할 경우의 에러의 종류와 코드 번호를 아래에 표시했다.

표 3 - 1 에러(ERROR)의 종류와 코드번호

코드번호	원 인	코드번호	원 인
0	FOR없는 NEXT	120	다시 선언된 배열
16	SYNTAX (문법)	133	영(ZERO)으로 나눔
22	GOSUB없는 RETURN	163	잘못된 형
42	DATA 부족	176	너무 긴 문자열
53	잘못된 수 값	191	복잡하게 된 공식
69	OVERFLOW (초과)	224	정의 안된 함수
77	메모리 부족	254	INPUT문에 잘못 응답함.
90	정의 안된 문	255	CTRL – C Interrupt Attempt
107	잘못된 첨자		

RESUME 에러처리에서 원하는 작업이 끝나면 RESUME에 의해 원래 에러가 생긴곳으로 다시 간다. 부프로그램의 끝인 RETURN문은 그 부프로그램을 부른 GOSUB한 바로 다음 행번호가 실행된 것과는 다소 다르다는 것을 명심하라.

다음은 수 A, B를 입력하여 A/B를 표시하는데 단 B가 0일때는 「0 DEVIDE」라고 표시하고 다시 입력하여 A가 0일때 끝나는 프로그램이다.

* PROGRAM 3.4

```
10  ONERR  GOTO 80
20  INPUT "A= ";A
30  IF A = 0 THEN  END
40  INPUT "B= ";B
50 AN = A / B
60  PRINT A;"/";B;"= ";AN
70  GOTO 20
80  PRINT "0 DEVIDE"
90 B = 1
100  RESUME
110  END
```

```
]RUN
A= 5
B= 3
5/3= 1.66666667
A= 7
B= 0
0 DEVIDE
7/1= 7
A= 0
```

행번호 50에서 B가 0이면 에러가 생겨 프로그램의 실행이 중지되고 경고가 나오는 것이 정상이지만 이와 같이 ONERR GOTO가 선언되면 그와 같은 일은 일어나지 않는다.

그러면 PEEK(222)에 의해 어떤 에러가 발생되었는가를 아는 프로그램의 예를 보자.

* PROGRAM 3.5

```
10  ONERR  GOTO 70
20 A = 100
30  FOR I = 100 TO  - 50 STEP  -
     20
40  PRINT A / I
50  NEXT I
60  END
70  PRINT "#"; PEEK (222);" ERROR
     CODE"
80 I =  - 10
90  RESUME
```

```
]RUN
1
1.25
1.66666667
2.5
5
#133 ERROR CODE
-10
-3.33333333
-2
```

이 프로그램은 A의 값을 100으로 하여 100, 80, 60, …, −50으로 나눗셈을 한 결과를 인쇄하는 것으로 행번호 10에 의해 0으로 나뉘는 때는 행번호 70으로 간다. 따라서 #133 ERROR CODE는 PRINT PEEK(222)에 의해 출력된 133은 결국 0으로 나뉘었다는 것을 뜻하고 있다.

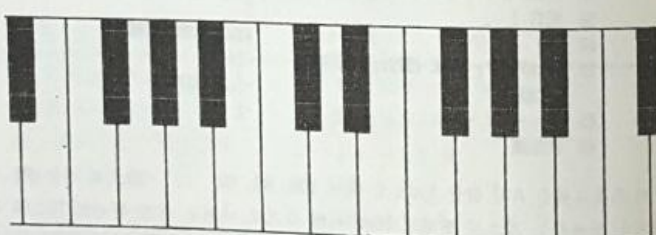
3-2 음악기능

* PROGRAM 3.6

```
0  REM  PIANO'S KEYBOARD
100  HOME : HGR : HCOLOR= 3
110  DATA  95,64,74,74,74,74,74,9
     5: FOR I = 8205 TO 15373 STEP
     1024: READ Y: POKE I,Y: NEXT
     I
120  DATA  38,41,41,41,105,41,41,
     38: FOR I = 8207 TO 15375 STEP
     1024: READ Y: POKE I,Y: NEXT
     I
130  DATA  1,1,1,127,0,8,8,127: FOR
     I = 8209 TO 15377 STEP 1024:
      READ Y: POKE I,Y: NEXT I
140  DATA     79,72,104,68,67,0,1
     ,127: FOR I = 8213 TO 15381 STEP
     1024: READ Y: POKE I,Y: NEXT
     I
150  DATA  41,41,111,41,47,0,1,12
     7: FOR I = 8215 TO 15383 STEP
     1024: READ Y: POKE I,Y: NEXT
     I
200  HPLOT 0,50 TO 270,50
210  HPLOT 0,150 TO 270,150
300  FOR X = 5 TO 270 STEP 20
310  HPLOT X,50 TO X,150
400  DATA     1,40,60,80,120,140,1
     80,200,220,260,1,1,1,1
405  READ Z
410  FOR I = Z TO Z + 10
430  HPLOT I,50 TO I,100
440  NEXT I
500  NEXT X

]RUN
```

피 아 노　건 반



애플의 스피커 출력은 번지를 한 번 액세스(access)할 때마다 한 개의 펄스가 나오는 원시적인 구조로 되어있기 때문에 펄스를 이용하여 그 펄스의 진동주기와 사이클의 수에 의해 음악을 연주할 수 있다.

따라서 컴퓨터 내부에 있는 기계어 코드와 프로그램을 수행하기 위한 메모리를 기억시키기 위해 CALL과 POKE를 함께 사용한 이 프로그램은 마치 기계어 코드로 쓰여진 프로그램과 같은 동작을 하게 된다.

이 프로그램은 음악을 연주하기위해 DATA문을 사용하게 되는데 당신은 이 과정을 알게되면 비교적 쉽게 컴퓨터의 스피커를 통해 음을 즐길 수 있을 것이다.

우리는 다음의 두행에 쓰여진 DATA문에 의해 기계어 코드를 BASIC으로 만들 수 있다. 물론 한 행으로 축소할 수 있지만 그것은 너무 길다.

```
10  HOME
100 DATA  173,48,192,136,208,5,2
    06,1,3,240,9
110 DATA  202,208,245,174,0,3,76
    ,2,3,96,0,0
```

메모리 할당에 있어서 시스템 이용 메모리 번지의 범위내에서 사용하게 되며, 일반적으로 메모리번지 768부터 1023까지 이용할 수 있다. 따라서 메모리번지 768과 769는 다른 목적에 사용하기 위해 메모리 번지 770부터 시작하도록 한다.

행번호 100과 110의 데이타가 전부 23개이므로 이것을 기억시키기 위해 메모리 번지 770에서 792까지 사용하게 된다.

```
120  FOR N = 770 TO 792
140  READ Y
160  POKE N,Y
180  NEXT N
```

이제 기계어코드에 대하여 모든 준비가 끝났다.

그러나 여기서는 다음의 두가지가 꼭 필요하다.

1. 남겨놓은 메모리 번지 768에 음정(pitch)을 설정하고 메모리 번지 769에는 음의 길이(duration)을 설정한다.

행번호 200과 220에서 음정과 음의 길이를 입력하도록 하고 그 입력된 데이타가 행번호 240과 260으로 기억된다.

```
200  PRINT : INPUT "WHAT PITCH (0
     -255) ";P
220  PRINT : INPUT "WHAT DURATION
      (0-255) ";D
240  POKE 768,P
260  POKE 769,D
```

2. 지금까지 DATA, READ, POKE을 사용해서 BASIC으로 기억시킨 이 프로그램을 사용하거나 실행시키기 위해서 행번호 300에 지정된 메모리 번지 770부터 실행하는 CALL 명령을 준다.

```
300  CALL 770
```

이들 명령문의 모임을 합쳐보면,

* PROGRAM 3.7

```
10  HOME
100 DATA  173,48,192,136,208,5,2
    06,1,3,240,9
110 DATA  202,208,245,174,0,3,76
```

```
    ,2,3,96,0,0
120 FOR N = 770 TO 792
140 READ Y
160 POKE N,Y
180 NEXT N
200 PRINT : INPUT "WHAT PITCH (0
    -255) ";P
220 PRINT : INPUT "WHAT DURATION
     (0-255) ";D
240 POKE 768,P
260 POKE 769,D
300 CALL 770
```

자, RUN을 타이프하고 RETURN 을 누르면 화면에는

```
WHAT PITCH (0-255)
```

가 나타날 것이다. 0과 255사이의 수치를 음정으로 정하여 타이프하고 RETURN 을 누르면 또 다시 화면에는

```
WHAT DURATION (0-255)
```

로 물어오는데 마찬가지로 음의 길이를 0에서 255 사이의 수치를 입력시키고 RETURN 를 누르면 이 음정과 음의 길이에 해당하는 음을 컴퓨터가 내게 된다.

그러면 이 기본적인 프로그램을 이용하여 음악을 연주해 보자.

먼저 음정을 0에서 255까지의 범위를 세분하여 음정표를 만들면 표3-2와 같다.

표3-2 음 정 표

	1st Octave	2nd Octave
A	228	114
B	204	102
C	192	96
D	171	85
E	152	76
F	144	72
G	128	64

여기서 가장 낮은 음정이 1st Octave의 「A」이고 가장 높은 음정은 2nd Octave의 「G」가 된다.

그리고 악보에 나타나는 모든 음정은 어떠한 틀에 속하는 것이 아니라, 그때 그때 악보를 보고 음정표의 각 음정을 설정하도록 한다. 물론 음의 길이도 0에서

255까지의 범위내에서 그때 그때의 악보의 박자를 고려하여, 분할하여 사용하기로 하자.

다음 악보를 컴퓨터에 의해 연주토록 프로그램을 짜 보자.

이 악보의 제목은 「The cliffs of Doneen」이다.

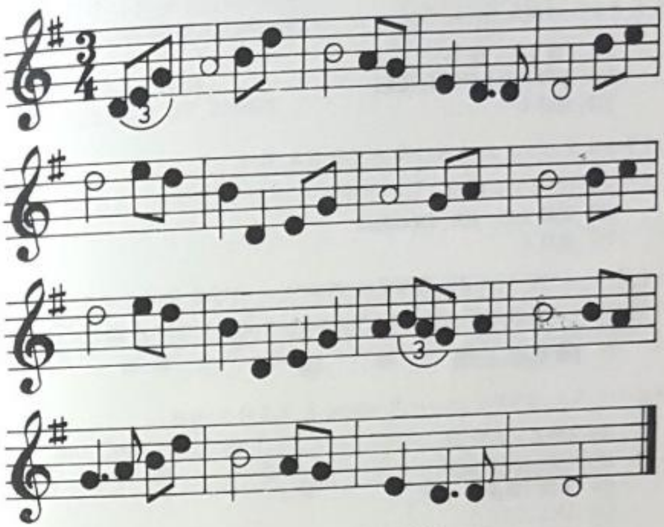


그림 3 - 3 The cliffs of Doneen

자, 음정과 음의 길이를 설정하면 각각의 데이타가 54개가 된다. 행번호 200에서 206까지의 54개의 데이타는 음정이다.

```
200  DATA  171,152,128,114,102,85
     ,102,114,128,152,171,171,171
202  DATA  102,96,85,76,85,102,17
     1,152,128,114,128,114,102
204  DATA  102,96,85,76,85,102,17
     1,152,128,114,102,114,128
206  DATA  114,102,102,114,128,11
     4,102,85,102,114,128,152,171
     ,171,171
```

다음 이 악보의 박자가 4 분의 3 박자이므로 한 박자를 120, 두 박자를 240으로 설정하여 54개의 음의 길이의 데이타를 행번호 220에서 226에 설정한다.

```
220  DATA  40,40,40,240,60,60,240
     ,60,60,120,180,60,240
221  DATA  60,60,240,60,60,120,12
```

```
      0,60,60,240,60,60,240
222  DATA  60,60,240,60,60,120,12
      0,120,120,120,40,40,40
226  DATA  120,240,60,60,180,60,6
      0,60,240,60,60,120,180,60,24
      0
```

먼저 음정의 배열을 P라고 하면

```
240  FOR C = 1 TO 54
260  READ P(C): REM  PITCHES
280  NEXT C
```

로 배열시킬 수 있고, 음의 길이의 배열은 D로 한다.

```
300  FOR C = 1 TO 54
320  READ D(C): REM  DURATIONS
340  NEXT C
```

그러면 이 데이타의 배열을 DIM문에 의해 먼저 선언해 두어야 하므로 각 데이타가 54개이므로 충분히 고려하여 60으로 해서 첫머리인 행번호 20으로 두면

```
20  DIM P(60),D(60)
```

여기에다 모든 음악루틴(Music Routine)을 호출하기 위해

```
360  FOR C = 1 TO 54
380  POKE 768,P(C)
400  POKE 769,D(C)
420  CALL 770
440  NEXT C
```

을 삽입한다.

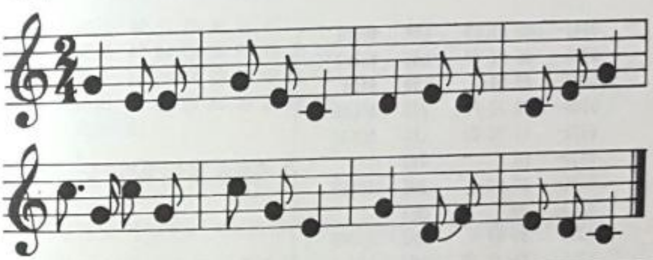
지금까지 첨가된 모든행을 중간중간에 설정된 프로그램의 원인을 쉽게 이해하기 위해 REM문을 삽입하여 프로그램을 완성하면 다음과 같다.

* PROGRAM 3.8

```
1  REM ************************
2  REM * MUSIC PROGRAM        *
3  REM *  THE CLIFFS OF DONEEN *
4  REM ************************
5  REM
10  HOME
20  DIM P(60),D(60)
99  REM  MACHINE COAD PROGRAM
100  DATA  173,48,192,136,208,5,2
      06,1,3,240,9
110  DATA  202,208,245,174,0,3,76
      ,2,3,96,0,0
120  FOR N = 770 TO 792
140  READ Y
160  POKE N,Y
180  NEXT N
199  REM  DATA FOR PITCHES OF NOT
      ES
200  DATA  171,152,128,114,102,85
      ,102,114,128,152,171,171,171
202  DATA  102,96,85,76,85,102,17
      1,152,128,114,128,114,102
204  DATA  102,96,85,76,85,102,17
      1,152,128,114,102,114,128
206  DATA  114,102,102,114,128,11
      4,102,85,102,114,128,152,171
      ,171,171
219  REM  DATA FOR DURATIONS OF N
      OTES
```

```
220  DATA  40,40,40,240,60,60,240
     ,60,60,120,180,60,240
221  DATA  60,60,240,60,60,120,12
     0,60,60,240,60,60,240
222  DATA  60,60,240,60,60,120,12
     0,120,120,120,40,40,40
226  DATA  120,240,60,60,180,60,6
     0,60,240,60,60,120,180,60,24
     0
240  FOR C = 1 TO 54
260  READ P(C): REM  PITCHES
280  NEXT C
300  FOR C = 1 TO 54
320  READ D(C): REM  DURATIONS
340  NEXT C
359  REM  ROUTINE TO PLAY TUNE
360  FOR C = 1 TO 54
380  POKE 768,P(C)
400  POKE 769,D(C)
420  CALL 770
440  NEXT C
```

문제 132. 다음의 「산토끼」의 악보를 컴퓨터에서 연주하는 프로그램을 작성하라.



```
1  REM ***********************
2  REM * MUSIC PROGRAM        *
3  REM *   SAN   TO   KKI     *
4  REM ***********************
5  REM
10  HOME
20  DIM P(30),D(30)
99  REM  MACHINE COAD PROGRAM
100  DATA  173,48,192,136,208,5,2
     06,1,3,240,9
110  DATA  202,208,245,174,0,3,76
     ,2,3,96,0,0
120  FOR N = 770 TO 792
140  READ Y
160  POKE N,Y
180  NEXT N
199  REM  DATA FOR PITCHES OF NOT
     ES
200  DATA  152,192,192,152,192,22
     8,204,192,204,228,192,152
210  DATA  114,152,114,152,114,15
     2,192,152,204,171,192,204,22
     8
219  REM  DATA FOR DURATIONS OF N
     OTES
220  DATA  240,120,120,120,120,24
     0,240,120,120,120,120,240
230  DATA  180,60,120,120,120,120
     ,240,240,120,120,120,120,240

240  FOR C = 1 TO 25
260  READ P(C): REM  PITCHES
280  NEXT C
300  FOR C = 1 TO 25
320  READ D(C): REM  DURATIONS
340  NEXT C
359  REM  ROUTINE TO PLAY TUNE
360  FOR C = 1 TO 25
380  POKE 768,P(C)
400  POKE 769,D(C)
420  CALL 770
440  NEXT C
```

> **문제** 133. 메모리 번지에 POKE명령을 써서 $ 300에는 D값(duration)을, $ 301 에는 F값(freguency)을 각각 넣고 $ 302부터 집행되는 것으로 하여, D 값과 F 값을 1에서 부터 255까지 차례로 변경시키면 소리를 내는 시간 D 값과 소리의 높낮이 F가 변화할 것이다.
> 낮은 음부터 시작하여 점점 높은 음으로 길이를 변화시키면서 높은 음에 이르면 끝내는 프로그램을 작성하고, 수행시켜보라.

이 경우, 일단 모니터 방식으로 소리를 내는 서브루틴을 기계어로 입력시켜야 한다. 해설에서는 기계어 뿐만 아니라 어셈블러 형태도 함께 출력시켰다.

```
]CALL-151                          ← BASIC 방식에서 모니터 방식으로 교체

*302L                              ← 모니터 방식의 LIST 명령

0302-   AC 01 03    LDY   $0301   ┐
0305-   AE 01 03    LDX   $0301   │
0308-   A9 04       LDA   #$04    │
030A-   20 A8 FC    JSR   $FCA8   │
030D-   AD 30 C0    LDA   $C030   │
0310-   E8          INX           │
0311-   D0 FD       BNE   $0310   ├ 소리를 내는 서브루틴
0313-   88          DEY           │
0314-   D0 EF       BNE   $0305   │
0316-   CE 00 03    DEC   $0300   │
0319-   D0 E7       BNE   $0302   │
031B-   60          RTS           ┘
        기계어      어셈블러
```

이때 기계어를 짜넣는 방법은 CALL-151을 타이핑하고 [RETURN]키를 누르면,

```
]CALL-151

*
```

이 화면에 나타나며, 커서가 깜박일 것이다. 그러면 이미 당신은 **모니터**(Monitor)를 사용할 수 있는 상태에 있다.

위의 문장은 사실상 151번지($ FF69)의 서브루틴을 부른 것으로써 모니터의 입구로 들어가는 기능을 수행한다.

그러면 다음과 같이 타이핑한다.

```
]CALL-151

*302:AC 01 03 AE 01 03 A9 04
```

```
*30A:20 A8 FC AD 30 C0 E8 D0

*312:FD 88 D0 EF CE 00 03 D0

*31A:E7 60
```

이때 연이어 타이프해도 좋다. 그러나 너무 길다고 생각하면 BASIC 방식에서와 같이 RETURN 키를 누르므로써 행이 바뀌면서 프롬프트 *가 나타난다. 이것을 타이프하는 방식은 처음의 메모리 번지를 16진수로 쳐 넣은 후 콜론(:)을 친 후 프로그램 기계어로 차례차례 키이를 두들겨 써 넣는다. 이 경우 주의할 것은 기계어와 기계어 사이는 한 칸씩 띄워 두어야 한다.

모두 타이핑한 후 바르게 타이프 했는지를 확인하기 위해서 다음과 같이 302. 320으로 메모리 번지 $ 302에서 $ 320까지의 메모리를 보게 된다.

```
*302.320

0302- AC 01 03 AE 01 03
0308- A9 04 20 A8 FC AD 30 C0
0310- E8 D0 FD 88 D0 EF CE 00
0318- 03 D0 E7 60 00 00 00 00
0320- 00
*
 00 00 00 00 00 00 00
```

이와같이 블럭단위로 검사하거나 확인하는 것을 덤프(dump)라 한다.

당신이 모니터 방식을 벗어나 다시 BASIC 방식으로 돌아가고 싶으면 CTRL - B 를 누른 뒤 RETURN 를 누른다. 이때 메모리에 있던 변수와 BASIC 프로그램은 모두 지워져 버린다.

지워지지 않게 하려면 CTRL — C 를 누른 후 RETURN 을 누른다. 그래서 CTRL — B 를 콜드스타트(cold start), CTRL — C 를 웜스타트(warm start)라 부른다.

다시] 모양의 BASIC 프롬프트 문자가 나타날 것이다.

```
10  FOR I = 1 TO 255
20 F = I
30 D = I
40  GOSUB 100
50  NEXT I
60  END
100  REM  --- SPEAKER
110  POKE 768,D
120  POKE 769,F
130  CALL 770
140  RETURN
```

자. 다음의 서브루틴으로 입력된 경우의 출력되는 음과 비교해 보아라.

```
]CALL-151

*302L

0302-   AD 30 C0    LDA   $C030
0305-   88          DEY
0306-   D0 05       BNE   $030D
0308-   CE 01 03    DEC   $0301
030B-   F0 09       BEQ   $0316
030D-   CA          DEX
030E-   D0 F5       BNE   $0305
0310-   AE 00 03    LDX   $0300
0313-   4C 02 03    JMP   $0302
0316-   60          RTS
```



3-3 시계기능

다른 기종의 컴퓨터와 달리 애플은 시계를 내장하고 있지 않다. 즉 TIME$ 혹은 TI$란 명령이 아예 없다.

그렇다고 불가능한 것은 아니다. 여기서는 여러분의 프로그램 작성에 도움을 주고자 미려하지는 않지만 시간의 흐름에 따라 숫자를 한 단계씩 증가시키므로써 시계의 구실을 할 수 있는 프로그램 작성 방식에 대해 설명하고자 한다.

FOR…NEXT문의 루프를 이용하므로 시간의 지연을 가져다 준다는 것은 앞에서 이미 설명하였다. 그러면 대체 애플에서는 몇 번 정도 루프가 돌때 1초 정도가 되는가? 일반적으로 712번의 루프가 1초가 된다. 자, 기본적인 「STOP-WATCH」 프로그램을 한번 보자.

```
* PROGRAM  3.9
1  REM
       STOP WATCH
2  REM
10 K = 712
20 N = 0
30 A =  INT (N / 60)
40 B = N - A * 60
100  FOR I = 1 TO K
110  NEXT I
120  PRINT  TAB( 5);A;":";B
130 N = N + 1
140  GOTO 30
200  END
```

```
]RUN
     0:0
     0:1
     0:2
     0:3
     0:4
     0:5
     0:6
     0:7
     0:8
     0:9
     0:10
     0:11
```

행번호 100에서의 I는 712번 반복되어 행번호 130에서 1을 더하여 초가 바뀌는

것을 나타낸다. 이 초(second)가 60이 되면 행번호 40에서 분으로 바뀌고 초는 0가 되어 다시 1부터 시작된다. 이 경우 프로그램을 RUN시킨 순간에 시간은 00분 00초에 맞추어지므로 이것은 스톱워치의 역할밖에 할 수 없다. 여기서 「00」이라는 의미를 한 번 생각해 보자.

00時 00分 00秒

시, 분, 초는 각각 2개의 단위로 연결된 6단위의 수로 구성되어 있다. 이 숫자를 입력시켜 앞의 프로그램인 「STOP WATCH」와 혼합시키면 그런대로 훌륭한 시계가 되지 않을까? 처음에 시간을 맞추기 위해 INPUT문을 이용해서 이런 식으로 작성해 보자.

```
10  INPUT T$
```

만일 당신의 시계가 현재 오전 10시 57분 12초라 한다면 키 입력을 다음과 같이 할 것이다.

```
105712
```

그러나 순간적으로 초의 입력이 불가능하므로

```
105800
```

으로 하고 약 48초를 기다렸다가 RETURN 키를 누르므로 컴퓨터의 시각도 맞게 될 것이다. 그럼 이 6단위의 수를 어떻게 시, 분, 초로 나누는 것인가가 문제이다. 이것은 문자함수를 사용하여 처리할 수 있을 것이다.

```
20 T1 =  VAL ( RIGHT$ (T$,2))
30 T2 =  VAL ( MID$ (T$,3,2))
40 T3 =  VAL ( LEFT$ (T$,2))
```

여기서 T1은 초, T2는 분, T3은 시를 나타낸다. 따라서 T1의 값은 루프방식에서 반복처리되므로 바뀌도록 하면 된다.

```
80  FOR I = 1 TO 712
90  NEXT I
```

이것은 수행하는 시간이 꼭 1초가 걸리므로 T1의 값에 의해 카운팅하게 된다. T1의 값이 60을 넘게 되면 행번호 60에서 T1은 0가 되면서 T2, 즉 분은 1이 더해지고 행번호 70은 분에서 시로 변하는 것을 나타내었고 행번호 75는 하루가 24시간이므로 23시 59분 59초 이후에는 0시0분0초가 되어야 하므로 24시가 될 때를 위해 첨가된 문이다. 만일 당신은 열차시각표와 같은 시간보다 통상의 시각을 읽는 방식으로 하고 싶을 때는 행번호 75를

```
75  IF T3 > 11 THEN T3 = T3 - 12:
        GOTO 75
```

으로 수정하면 된다. 이것을 종합하면 프로그램 「WATCH」가 된다.

```
* PROGRAM 3.10
 1  REM ************************          0:0:1
 2  REM *   W A T C H          *          0:0:2
 3  REM *     PROGRAMED BY     *          0:0:3
 4  REM *      KWACK SUNG KEUN *          0:0:4
 5  REM ************************          0:0:5
 6  REM
 10  INPUT T$                          ]RUN
 20 T1 =  VAL ( RIGHT$ (T$,2))         ?110057
 30 T2 =  VAL ( MID$ (T$,3,2))            11:0:57
 40 T3 =  VAL ( LEFT$ (T$,2))             11:0:58
 50  GOTO 100                             11:0:59
 60  IF T1 = 60 THEN T1 = 0:T2 = T        11:1:0
     2 + 1                                11:1:1
 70  IF T2 = 60 THEN T2 = 0:T3 = T        11:1:2
     3 + 1                                11:1:3
 75  IF T3 > 23 THEN T3 = T3 - 24:        11:1:4
      GOTO 75
 80  FOR I = 1 TO 712                  ]RUN
 90  NEXT I                            ?035900
 100  PRINT  TAB( 5);T3;":";T2;":"        3:59:0
      ;T1                                 3:59:1
 110 T1 = T1 + 1                          3:59:2
 120  GOTO 60                             3:59:3
                                          3:59:4
]RUN                                      3:59:5
?235958                                   3:59:6
    23:59:58                              3:59:7
    23:59:59                              3:59:8
    0:0:0
```

만일 이 프로그램에 의한 당신의 컴퓨터에서 시간의 차이가 있다면 행번호 80의 712를 수정하면 될 것이다. 예로써 시간이 늦으면 712보다 작은 값으로, 시간이 빠르면 712보다 큰 값을 주므로써 해결되리라 본다.

3-4 특수번지의 사용

이 절에서 설명하고자 하는 것은 PEEK, POKE, CALL 명령 혹은 문에 의해 사용할 수 있는 애플소프트 BASIC의 특별한 기능이다.

메모리에는 번지가 지정되어 있어 BASIC으로 프로그램을 작성하면 자동적으로 프로그램의 내용이나 데이타가 메모리의 적절한 번지에 기억된다. BASIC의 특징은 프로그램 작성시에 이에 신경을 쓰지 않아도 된다는 점이다. 애플은 애플소프트 프로그램을 $800번지의 메모리부터 기록하기 시작한다. 메모리의 지정

된 번지에 어떤 내용을 격납할 때는 POKE, 그 내용을 알아내고 싶을 때는 PEEK라는 명령어를 사용한다.

간단한 스위칭의 경우는 일반적으로 번지만이 관계한다. 다시말하면 그 번지가 액세스됨으로써 스위칭이 일어나는 경우가 많다. 예를 들면

```
]POKE -16304,0
```

이라는 스위칭은

```
]X=PEEK(-16304)
```

와 꼭같은 값이다. 개인용 컴퓨터(personal computer)는 대부분 8비트용 중앙처리 장치를 쓰고 있으므로 메모리 번지는 $0000(0)에서 $FFFF(65535) 번지까지 쓸 수 있다.

여기서 PEEK(N)은 N번지의 내용을 알고자 하는 것으로 N은 항상 10진수여야 한다. 이때 32767 보다 큰 수는 보수를 사용하여 나타낼 수 있다. 32767보다 큰 번지를 N이라고 하면 그 보수 X는

```
X = N - 65536
```

으로 나타난다. 따라서 위의 예에서 N의 값은 49232가 된다.

3-4-1 제로페이지 (ZERO PAGE)

애플의 마이크로 프로세서 6502는 65536개의 메모리를 가질 수 있다. 보통256개의 메모리번지로 된 메모리를 페이지라고 부르며, 64K바이트일 경우 0페이지부터 시작하여 255페이지가 된다고 보면 편리하다.

따라서 당신이 특별히 알아야 할 메모리 내용은 제로 페이지(ZERO PAGE: 0 page)로 불리는 $00에서 $FF번지 사이의 내용이다.

그림3-4는 애플의 시스템 메모리 페이지의 각 용도를 나타내었다.

텍스트 윈도우 (text window)

텍스트윈도우(text window)란 화면상에서 스크롤(scroll)이 일어나고 있는 범위를 가리킨다. 애플 소프트 BASIC에서는 보통 윈도우는 풀 스크롤(full scroll)에 세트되어 있는데, 제로 페이지의 특정한 번지에 특정한 값을 POKE로 써 넣으면, 이 윈도우의 크기를 변화시킬 수 있다.

텍스트 윈도우를 세트해도 커서는 자동적으로 윈도우내로 들어가지 않는다. 따라서, 윈도우를 세트한 후는 HOME, TAB, VTAB, HTAB등을 사용해서 커서를 윈도우 안에 넣어주지 않으면 안된다. 한번 윈도우 속으로 커서가 들어가면 커서는 지정된 범위 속에서 밖에 움직이지 않게 된다.

VTAB는 윈도우를 무시하고 움직일 수 있다. 윈도우 위에 인쇄된 행은 정상이나, 윈도우보다 아래에 인쇄될 경우에는 포맷은 무시되고 모든 인쇄는 1행씩 밀려간다. HTAB도 커서를 윈도우 밖으로 내놓을 수 있는데, 1문자를 인쇄하면 커서는 윈도우 속으로 되돌아간다.

페 이 지 (page)		용 도
10 진 수	16 진 수	
0	$ 00	제로페이지
1	$ 01	스택
2	$ 02	GETLN 입력버퍼
3	$ 03	벡터
4 7	$ 04 $ 07	텍스트 및 저해상도 그래픽(GR)
8 11	$ 08 $ 0B	텍스트 및 저해상도 그래픽(GR)
12 31	$ 0C $ 1F	사용자가 사용가능한 RAM
32 63	$ 20 $ 3F	고해상도 그래픽(HGR)
64 95	$ 40 $ 5F	고해상도 그래픽(HGR 2)
96 191	$ 60 $ BF	디스크 운영체제
192 207	$ C0 $ CF	I/O
208 255	$ D0 $ FF	ROM

그림 3 - 4 시스템 메모리map

윈도우를 변경하면 그것은 컴퓨터에 써넣기를 하는 메모리 번지를 변경하는 것을 나타낸다. 윈도우를 결정하는 패러미터가 어느 범위에 들어가 있을 경우 문제는 없지만, 허용되지 않는 값을 사용하면 매우 곤란한 사태로 되는 수가 많다.예로써 텍스트 좌단의 값을 255로 하면, 텍스트는 최대 40문자가 1행이므로 좌단은 0에서 40까지의 값을 취하게 된다. 이것은 애플에 화면 메모리의 범위를 훨씬 넘은 번지에 써 넣기를 행하는 것을 나타낸다. 따라서 프로그램을 지키기 위해서는 텍스트와 윈도우의 패러미터는 허용된 값의 범위내에 있어야 한다.

```
10 POKE 32,L
```

이것은 윈도우의 좌단의 위치를 L로 지정한 값에 세트한다. 따라서 이 번지의 값은 0부터 39까지의 값을 가질 수 있으며 보통때는 화면이 가장 왼쪽부터 시작하므로 L은 0가 된다.

```
20 POKE 33,W
```

33($21) 번지에는 화면의 가로폭이 들어있어 W에서 지정한 값에 세트한다. 즉 W는 1부터 40의 값을 취할 수 있다. 단, 윈도우의 폭 W는 절대로 0으로 할수 없다. 애플소프트가 폭주한다.

위의 $20(32), $21(33)의 예는 반드시 쌍으로 해서 사용한다. 우선 폭 W를

세트하고, 그 다음 텍스트 윈도우의 가로 시작점 L의 값을 세트하는 편이 좋을 것이다. 여기서 W가 33이하 일때, PRINT문의 3개째의 tab field는 윈도우 밖에 캐릭터 출력을 행할 수 있다.

```
30 POKE 34,T
```

윈도우의 상단을 지정한다. T는 0에서 23의 범위에서 가능하며, 다음에 설명하는 하단 제어보다 큰 값으로 해서는 안된다.

```
40 POKE 35,B
```

이 번지 35($23)에는 텍스트화면의 최하단 위치를 결정하는 값이 들어 있다. B는 0에서 23의 값을 취할 수 있으나, 상단보다 작은 값으로 해서는 안된다.

이것을 정리하여 텍스트 윈도우의 범위 값을 보면

$0 \leq L \leq 39$
$1 \leq W \leq 40$
$0 \leq T \leq 23$
$0 \leq B \leq 23$
$T \leq B$
$L + W \leq 40$

가 된다.

이상의 $20(32), $21(33), $22(34), $23(35)의 4가지 번지는 텍스트 윈도우를 결정할 때 많이 사용되며 텍스트화면내에서 스크롤이나 커서의 이동도 이 4가지 위치로 결정된 윈도우 내에서만 가능하다.

예를들면 그림 3-5와 같이 윈도우를 설정하려면

```
]POKE 32,10:POKE 33,30:POKE 34,5:POKE 35,20
```

를 실행시키면 된다. 이들 번지들을 조작할때 주의할 점은 커서가 반드시 새로 정한 윈도우내에 들어와 있어야 한다는 점이다.

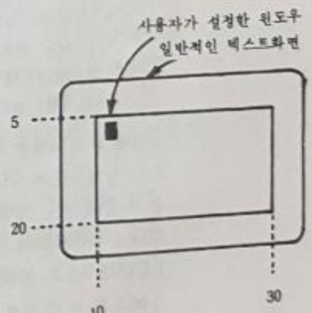


그림 3-5 텍스트 윈도우

커서에 관한 명령

```
50 CH = PEEK (36)
```

36($24)번지에는 커서의 현존의 수평위치가 들어있고, 위의 예에서 CH에 그 값이 들어간다. 윈도우가 지정되어 있지 않을 때에는 CH는 0에서 39까지의 값을 가질 수 있으며, 지정되어 있으면 윈도우의 좌단에서 본 상대적인 값을 돌려온다.

```
60 POKE 36,CH
```

CH 에서 주어진 위치에 커서를 가지고 간다. 화면의 좌단은 0 이므로, 실제의 위치로서는 CH+ 1 번째가 된다.

```
70 CV = PEEK (37)
80 POKE 37,CV
```

37 ($25) 번지에는 커서의 수직위치가 절대위치로 들어가 있다. CV = 0 은 제일 위의 행이고, CV=23은 제일 아래의 행을 나타낸다. 따라서 행번호 80과 같이 하면 CV의 위치로 커서가 옮긴다.

BASIC 프로그램의 메모리 기록 방식

```
]CALL-151

*67.68
```

메모리번지 $ 67 (103) 과 $ 68 (104) 에는 애플 소프트 BASIC프로그램이 시작되는 번지를 지정한다. 따라서 LOMEM값과 일치하며 일반적으로 시작번지는 $801번지가 된다.

```
0067- 01
0068- 08
```

여기서 $67에는 시작번지의 하위바이트이고 $68에는 상위바이트를 나타낸다. 그리고 $69 (105) 와 $6A (106) 번지에는 마지막 메모리 바이트에 관한 정보를 저장한다.

그러면 다음과 같은 간단한 애플소프트 BASIC프로그램을 보자.

```
* PROGRAM 3.11
 1 REM APPLE 0 PAGE
 2 PRINT "0"
 3 END
```

이제 프로그램을 체크하기 위해 모니터로 들어가 보자. BASIC 서브루틴을 부르는 방법가운데 GOSUB라는 명령어가 있다. 어셈블리 언어나 기계어 프로그램을 부를때 쓰는 명령어는 CALL이다. 만일 기계어 서브루틴이 N 번지에서 시작된다면 GOSUB와 같이 CALL N으로 하면 된다. GOSUB로 부른 서브루틴은 RETURN으로 원래 위치로 되돌아 오지만 CALL로 부른 서브루틴은 RTS 또는 $60으로 원래 위치로 점프하게 된다.

다시 모니터로 들어가자.

```
]CALL-151
*67.68
0067- 01
0068- 08
*69.6A
0069- 26 08
```

여기서 마지막 메모리 바이트는 $826이다. 다음을 타이핑하자.

```
*800.826

0800- 00 00 00 01 00 B2 20 41
0808- 50 50 4C 45 20 30 20 50
0810- 41 47 45 00 1D 08 02 00
0818- BA 22 30 22 00 23 08 03
0820- 00 80 00 00 00 FF FF
```

이 리스트가 앞에서의 BASIC프로그램의 메모리 덤프이다. BASIC 프로그램이 위의 16진수 덤프로 어떻게 번역하는가를 살펴보자.

$800번지의 $00은 아무 특별한 의미가 없다. 사실상 $800번지는 항상 $00을 포함한다. 이제부터 프로그램의 행이 시작된다. 각 행에는 4 바이트로 지정된다. 처음 2 바이트는 다음행의 시작바이트 번지를 저장한다. 즉 위의 예에서 $801과 $802번지에서 다음행이 $814메모리 번지에서 시작됨을 표시한다. 두번째 한쌍의 바이트는 행번호를 표시한다. 하위바이트가 $01이고 상위바이트가 $00이므로 $0001이므로 행번호는 1이다. 따라서 $803과 $804메모리 번지는 첫번째 행번호 1을 표시한다. 각 행은 끝에 다음행과 구분하기 위해 「00」이란 하나의 바이트를 삽입한다. 따라서 프로그램의 한 행은 머리 부분만도 5 바이트가 필요하다. 한 행에 콜론(:)으로 여러개의 문을 묶으면 추가 행을 사용할 때보다 4 바이트가 절약하게 된다. 1 바이트는 콜론을 나타내어야 하기 때문이다. 두번째 행은 $814부터 시작된다.

부록C 의 중간언어는 애플에서 사용되는 모든 명령어의 16진법, 10진법 리스트이다. 이를 **토큰**이라 한다. 예를 들면 $805의 $B2는 REM, END의 토큰은 $80, PRINT의 토큰은 $BA이다. $806의 $20은 SPACE인데 이것은 ASCII 코드에 따른다. 다음의 $41은 A, $50은 P, 또 $50은 P, $80A 의 $4C는 L 이고 $80B의 $45는 E 이다. ASCII코드에 의해 처음 시작한 BASIC 프로그램과 비교해 보라.

마지막으로 강조할 점은 당신이 프로그램을 END문으로 끝냈다 할지라도 컴퓨터는 이것이 프로그램의 끝임을 알지 못한다는 점이다. 그러나 프로그램의 끝은 다음 행 번호가 들어갈 위치에 「00 00」이 들어감으로써 표시된다.

```
100 X = PEEK (218) + PEEK (219)
    * 256
```

에러코드

ONERR-GOTO문이 사용되고 있을때, 에러가 발생하면 $DA(218), $DB(219) 번지에는 에러가 일어난 행의 행번호가 저장된다. 따라서 예와 같이 하면 X에는 그 행번호가 주어지게 된다.

```
110 IF PEEK (216) > 217 THEN GOTO
    1000
```

$D8 (216) 번지의 비트 7 이 1 인 것은 ONERR GOTO문이 이미 실행된 것을

나타내고 있다.

```
120  POKE 216,0
```

이것은 ERRFLG(에러플랙)을 클리어하고 보통의 에러 메세지가 표시되도록 하는 명령이다.

```
130 Y =  PEEK (222)
```

$DE(222) 번지에는 ONERR문이 작용하고 있을때 발생한 에러코드가 기억되어 있다. 에러의 종류를 나타내는 에러 코드의 값을 Y로 하여 에러의 종류를 살펴보면 다음과 같다.

(Y의 값)

코드	에러의 종류	코드	에러의 종류
0	NEXT WITHOUT FOR ERROR	22	RETURN WITHOUT GOSUB ERROR
1	LANGUAGE NOT AVAILABLE ERROR	42	OUT OF DATA ERROR
2,3	RANGE ERROR	53	ILLEGAL QUANTITY ERROR
4	WRITE PROTECTED ERROR	69	OVERFLOW ERROR
5	END OF DATA ERROR	77	OUT OF MEMORY ERROR
6	FILE NOT FOUND ERROR	90	UNDEFINED STATEMENT ERROR
7	VOLUME MISMATCH ERROR	107	BAD SUBSCRIPT ERROR
8	I/O ERROR	120	REDIMENSION ARRAY ERROR
9	DISK FULL ERROR	133	DIVISION BY ZERO ERROR
10	FILE LOCKED ERROR	163	TYPE MISMATCH ERROR
11	SYNTAX ERROR (DOS)	176	STRING TOO LONG ERROR
12	NO BUFFER AVAILABLE ERROR	191	TOO COMPLEX ERROR
13	FILE TYPE MISMATCH ERROR	224	UNDEFINED FUNCTION ERROR
14	PROGRAM TOO LARGE ERROR	254	BAD INPUT RESPONSE ERROR
15	NOT DIRECT COMMAND	255	CTRL-C INTERRUPT ERROR
16	SYNTAX ERROR (APPLE SOFT)		

예를 들면

* PROGRAM 3.12

```
10  ONERR  GOTO 30
20  NEXT
30  PRINT "ERROR CODE= "; PEEK (2
    22)

]RUN
ERROR CODE= 0
```

을 실행시키면 Y의 값이 0가 되어 NEXT문에 대응되는 FOR가 없어 에러가 발생했다는 것을 의미한다.

3-4-2 키보드 및 텍스트 윈도우 제어기능

```
50 CALL - 936
```

-936번지에서는 화면을 클리어해서 커서를 좌측상단으로 가지고 오는 루틴이 들어 있다. 이 CALL은 ESC SHIFT - @/P 나 「HOME」명령과 같은 작용을 한다.

```
60 CALL - 958
```

현재의 커서위치에서 뒤를 향해 모든 화면과 윈도우 안에서 클리어한다. 커서보다 좌측 또는 커서보다 위에 있는 캐릭터는 영향을 받지 않는다. ESC F와 같은 작용을 한다.

```
70 CALL - 868
```

커서에서 우측의 화면을 1행만 클리어한다. ESC E 와 같다.

```
80 CALL - 922
```

라인피드를 일으킨다. CTRL-J와 같다.

```
90 CALL - 912
```

텍스트를 윈도우내에서 일단 스크롤시킨다.

```
100 X = PEEK ( - 16384)
```

키보드의 포트를 읽어 넣는다. 키가 눌러지면 X의 값은 127보다 커진다. X의 값은 ASCII코드에 비트 7을 더한 것과 같아진다.

```
110 POKE - 16368,0
```

키보드의 스토로우브를 클리어 한다. 이 조작은 키보드를 해독한 직후에 반드시 시행할 필요가 있고, 클리어해 두지 않으면 다음 문자를 해독할 수 없게 된다.

3-4-3 게임 컨트롤 및 스피커에 관한 명령

```
150 X = PEEK ( - 16336)
```

스피커를 1회만 울린다. 이 때에는 스피커에서는 「딸까닥」하는 소리가 들릴 뿐이다.

```
160 X = PEEK ( - 16352)
```

카세트의 출력은 1회만 울린다.

```
170 X = PEEK ( - 16287)
```

이것은 패들 # 0에 붙어 있는 버튼의 상태를 조사한다. 버튼이 눌려졌을 때에는 X의 값이 127보다 커진다.

```
180 X =  PEEK ( - 16286)
190 X =  PEEK ( - 16285)
```

행번호 180은 게임패들 # 1의 버튼의 상태를, 행번호 190은 게임패들 # 2의 버튼의 상태를 조사한다.

```
200  POKE  - 16296,1
```

게임코넥터의 「annunciater」 출력# 0을 TTL오우픈 콜렉터 출력 high로 한다. 이 상태는 OFF상태이다.

```
210  POKE  - 16295,0
```

게임출력 # 0을 TTL low한다. 이 상태는 ON의 상태로 최고 1.6mA의 출력 전류를 흘린다.

```
220  POKE  - 16294,1
230  POKE  - 16293,0
```

게임출력 # 1을 TTL을 행번호 220은 high, 행번호 230은 low로 한다.

```
240  POKE  - 16292,1
250  POKE  - 16291,0
```

게임출력 # 2을 TTL high, low로 한다.

```
260  POKE  - 16290,1
270  POKE  - 16289,0
```

행번호 260은 게임출력 # 3을 TTL high, 행번호 270은 TTL low로 한다.

참고

제로페이지 (ZERO PAGE)의 특수번지

16 진	10 진	내 용
$ 0 ~ $ 5	0 ~ 5	시스템 재기동용 점프로케이션
$ A ~ $C	10~12	USR 함수용 점프명령
$ D ~ $ 17	13~23	다목적 카운터 1플랙
$ 20	32	텍스트 윈도우의 좌측상단 시작점
$ 21	33	텍스트 윈도우의 폭
$ 22	34	텍스트 윈도우의 세로시작점
$ 23	35	텍스트 윈도우의 하단 끝점
$ 24	36	커서의 수평위치
$ 25	37	커서의 수직위치
$ 2B	43	부트 슬로트
$ 2C	44	최후의 HLIN, VLIN 또는 PLOT의 위치
$ 30	48	저해상도에서의 색지정
$ 32	50	텍스트출력 포맷
$ 33	51	프롬프트 문자
$ 4A, 4B	74~75	LOMEM 번지 (Intgger BASIC)

16 진	10 진	내 용
$ 4C, 4D	76~77	HIMEM번지 (Integer BASIC)
$ 4E, 4F	78~79	난수 피일드
$ 62~66	98~102	최후에 행한 곱셈 또는 나눗셈의 결과
$ 67, 68	103~104	프로그램의 시작번지. 보통 firmware 버전에서는 $801. 카세트 버전에서는 $3001에 자동적으로 세트된다.
$ 69, 6A	105~106	변수공간 시작번지
$ 6B, 6C	107~108	배열공간 시작번지
$ 6D, 6E	109~110	수변수 영역 최종번지
$ 6F, 70	111~112	스트링 영역 시작번지 스트링 변수는 여기서 HIMEM을 향해 격납된다.
$ 71, 72	113~114	다목적 포인터
$ 73, 74	115~116	HIMEM 번지 (Apple soft BASIC)
$ 75, 76	117~118	현재 실행중의 행번호
$ 77, 78	119~120	END/STOP문 또는 CTRL-C 에 의해 중단 되었을 때 최후에 실행하던 행번호
$ 79, 7A	121~122	수행중인 행번호
$ 7B, 7C	123~124	현재 읽어넣기 중인 DATA문의 행번호
$ 7D, 7E	125~126	DATA가 위치하는 번지
$ 7F, 80	127~128	INPUT문에 대한 입력선
$ 81, 82	129~130	최후에 사용된 변수명
$ 83, 84	131~132	최후에 사용된 변수번지
$ 85~9C	133~156	다목적 작업영역
$ 9D~A3	157~163	부동 소수점 연산용, 주어큐뮤레이터
$ A4	164	부동소수연산 루틴용 다목적 포인터
$ A5~AB	165~171	부동소수점 연산용, 부어큐뮤레이터
$ AC~AE	172~174	다목적 포인터/플랙
$ AF, B0	175~176	프로그램의 종료번지
$ CA, CB	202~203	프로그램의 시작번지 (Integer BASIC)
$ CC, CD	204~205	변수 저장의 최종번지 (Integer BASIC)
$ D6	214	RUN플래그
$ D8	216	ONERR 플래그
$ DA, DB	218~219	ONERR가 발생한 행
$ DE	222	ONERR 에러 코드
$ ED, E1	224~225	최후의 HPLOT X좌표
$ E2	226	최후의 HPLOT Y좌표
$ E4	228	HCOLOR 값
$ E6	230	고해상도 페이지
$ E7	231	SCALE값
$ E8, E9	232~233	도형정의표 시작번지
$ EA	234	고해상도 그래픽스, 카운터
$ F1	241	SPEED값
$ F3	243	플래시 마스크
$ F4~F8	244~248	ONERR문용 포인터
$ F9	249	ROT값

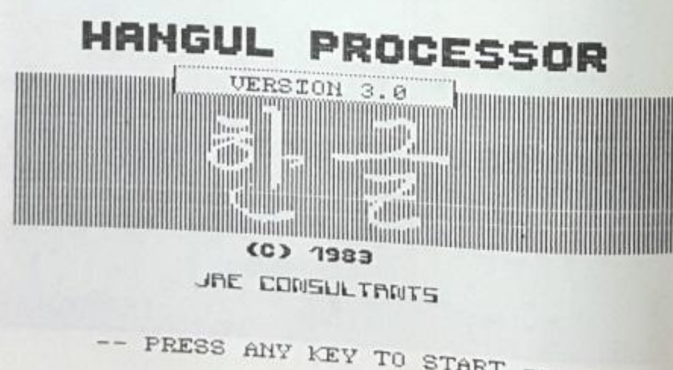
3－5 한글표시기능

한글을 표시할 수 있는 APPLE Ⅱ K-PLUS 에서는 메인보드 및 모니터ROM을 약간 변경하여 한글을 표시할 수 있게 하였다. 키 배열은 영·한글 혼용 타이프라이터와 거의 같다. 한글을 나타내는 코드는 기본적으로 inverse 모드에 할당된 문자코드 부분을 소프트웨어 스위치로 바꾸는 것으로 한글⇌inverse 표시를 바꿀 수 있다. 따라서 동일화면 내에 한글과 inverse 표시가 함께 사용될 수 없다.

표 3－3은 화면 표시의 문자코드표로 한글을 inverse 문자와 변경한 것이다.

표 3－3 화면표시의 charator code표

Decimal	Hex	0 $00	16 $10	32 $20	48 $30
0	$0	@	P (ㅐ)		0
1	$1	A (ㅁ)	Q (ㅂ)	!	1
2	$2	B (ㅠ)	R (ㄱ)	"	2
3	$3	C (ㅊ)	S (ㄴ)	#	3
4	$4	D (ㅇ)	T (ㅅ)	$	4
5	$5	E (ㄷ)	U (ㅕ)	%	5
6	$6	F (ㄹ)	V (ㅍ)	&	6
7	$7	G (ㅎ)	W (ㅈ)	'	7
8	$8	H (ㅗ)	X (ㅌ)	(	8
9	$9	I (ㅑ)	Y (ㅛ)	)	9
10	$A	J (ㅓ)	Z (ㅋ)	* (ㅃ)	: (₩)
11	$B	K (ㅏ)	[	+	; (ㅖ)
12	$C	L (ㅣ)	\	, (ㅒ)	< (ㅆ)
13	$D	M (ㅡ)	]	- (ㄲ)	= (ㄸ)
14	$E	N (ㅜ)	^	.	> (ㅉ)
15	$F	O (ㅔ)	_	/	?



한글은 초성+중성+종성 혹은 초성+중성으로 구성된다. 따라서 한글은 가로 14도트, 세로로 16도트를 할당하므로 한글은 가로 20문자 12행으로 화면에 나타낼 수 있다.

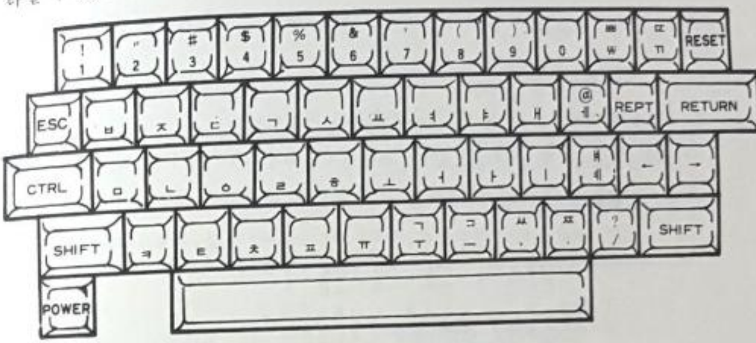


그림 3 - 6 한글의 키보드 및 배열

만약 당신이 디스크 드라이브와 한글 디스켓이 있다면 한글 디스켓을 디스크드라이브(사용법 4장참조)에 넣고 PR# 6 를 타이프한다.
그러면 몇초간 시끄러운 음을 내면서 디스크 드라이브가 부트(boot)한다. 디스크가 회전을 멈추면 다음과 같은 메세지가 표시될 것이다.

```
당신의 DISKETTE을 넣고 RETURN을 치시오
```

당신이 앞으로 한글 프로그램을 짜서 저장할 디스켓을 한글 디스켓과 바꾸어 넣는다. 다시 디스크가 가동하는 소리가 들리고 이어 프롬프트「]」와 커서「-」가 표시된다.

그러면 예로써

```
10  PRINT "
```

를 치고 CTRL-K를 누르므로 한글 기능을 표시할 수 있다. 그림3-6의 키보드를 참조하여「ㅎㅏㄴㄱㅡㄹ」이라고 타이핑하고 스페이스바를 누르고 다시「ㄱㅣㄴㅡㅇ」을 타이프 한다. 다음은 CTRL-A를 누른다. 이 기능은 한글모드에서 알파벳으로 변환하는 명령이다. 주의할 것은 한글 표시는 아직 그래픽으로 되어 있으므로 따옴표 내에서만 가능하다.

* PROGRAM 3.13

```
10  PRINT "한글 기능"

]RUN
한글 기능
```

그런데 프로그램을 LIST할 때 모든 한글 표시가 알파벳으로 되어 있을 경우가 있다. 예로써 「한글 기능」이라는 한글 표시가

```
GKSRMF RLSMD
```

으로 나타난다. 이때

```
]CALL3327
```

를 직접실행 명령으로 주게 되면 한글기능이 되살아난다.

다음의 프로그램을 보자.

* PROGRAM 3.14

```
10 A$ = "쥐,소,호랑이,토끼,용,뱀,말,양,
      원숭이,닭,개,돼지"
20 B$ = LEFT$ (A$,29)
30  PRINT B$

]RUN
쥐,소,호랑이,토끼,용,뱀,
```

위의 프로그램은 왼쪽에서 29번째까지 PRINT 하는 것으로 한글의 한 문자가 여기에서는 한문자로 취급된 것이 아님을 알 수 있다. 다음을 첨가해 보자.

```
40 C$ = RIGHT$ (A$,32)
50  PRINT C$
```

이것은 오른쪽에서 32번째의 문자로부터 끝까지 PRINT하는 것으로 지금까지의 프로그램을 종합하여 수행시키면 다음과 같다.

* PROGRAM 3.15

```
]LIST
10 A$ = "쥐,소,호랑이,토끼,용,뱀,말,양,
      원숭이,닭,개,돼지"
20 B$ = LEFT$ (A$,29)
30  PRINT B$
40 C$ = RIGHT$ (A$,32)
50  PRINT C$

]RUN
쥐,소,호랑이,토끼,용,뱀,
말,양,원숭이,닭,개,돼지
```

한글을 이용한 스트링 변수로 문장을 만들어 보자.

* PROGRAM 3.16

```
]LIST
10 A$ = "당신":B$ = "나."
```

```
20 C$ = "는":D$ = "을"
30 E$ = "사랑해"
40  PRINT B$;C$;" ";A$;D$;" ";E$
]RUN
나는 당신을 사랑해
```

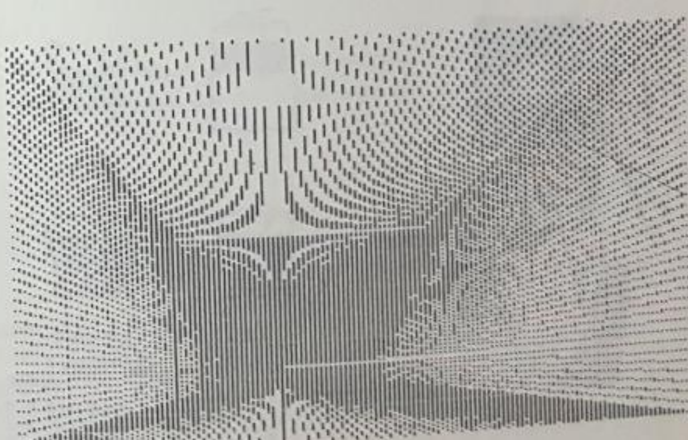
> **문제 134.** 「학년」, 「반」, 「학생수」, 「과목수」, 「과목당 만점」을 입력하여 「과목명」과 각 학생의 「성적」을 차례로 넣어 성적을 처리하는 프로그램을 작성하라.

```
45  HTAB 40: VTAB 24
50  GET A$
60   & : VTAB 3: PRINT "다음사항을 넣어주십시오 "
70   INPUT "학년 ";Y: INPUT "반 ";C
75   INPUT "학생수 ";N: INPUT "과목수 ";B
80   INPUT "과목당 만점 ";F
90  DIM A(N - 1,B + 3),BK$(B - 1)
95   & : PRINT "과목명을 차례대로 넣어주십시오 "
97  VTAB 4
98  FOR I = 0 TO B - 1: INPUT "";BK$(I): NEXT
100   & : VTAB 2: HTAB 3: PRINT "지금부터 각 학생의 성적을
      차례로 넣어주십시오 "
110   PRINT "준비가 되었^으면 아무 키나 눌러주십시오 "
120  GET A$
130  &
134  GOSUB 135: GOTO 190
135 H = 1:V = 1
140  FOR I = 0 TO B - 1
150  POKE 36,(H - 1) * 10 + 1: POKE 37,V
160  PRINT BK$(I);:H = H + 1: IF H = 5 THEN 170
165  GOTO 180
170 H = 1:V = V + 2
180  NEXT I
185  RETURN
190  FOR MK = 0 TO N - 1
193 SUM = 0:AVR = 0
195 H = 1:V = 1
198  PRINT
200  FOR J = 0 TO B - 1
205  POKE 36,(H - 1) * 10 + 6: POKE 37,V
210 S$ = ""
220  GET A$: IF A$ = ":" THEN 250
230  PRINT A$;:S$ = S$ + A$: GOTO 220
250 A(MK,J) =  VAL (S$)
255 SUM = SUM + A(MK,J)
260 H = H + 1: IF H = 5 THEN 300
270  GOTO 310
300 H = 1:V = V + 2
305  PRINT
310  NEXT J
320  POKE 36,1: POKE 37,22: PRINT "C/N?";
330  GET D$: IF D$ = "N" THEN 350
340  GOTO 370
350   POKE 36,1: POKE 37,22: PRINT "과목번호와 성적 "
355  INPUT T,R
360 SUM = SUM - A(MK,T - 1) + R:A(MK,T - 1) = R: GOTO 320
370  & : GOSUB 135
380 A(MK,B) = SUM:A(MK,B + 1) =  INT (SUM / B * 100) / 100
400  NEXT MK
600  FOR TT = 0 TO N - 1
610 A(TT,B + 3) = A(TT,B):A(TT,B + 2) = TT + 1
620  NEXT TT
```

```
2030 K = 0:L(K) = 0:R(K) = N - 1
2040  IF K >  = 0 THEN 2060
2050  GOTO 2430
2060 LL = L(K):RR = R(K):K = K - 1
2070  IF RR - LL >  = MM THEN 2090
2080  GOTO 2280
2090 I = LL:J = RR:MM = A(I,B + 3):WW = A(I,B + 2)
2100  IF MM < A(J,B + 3) THEN 2120
2110  GOTO 2130
2120 J = J - 1: IF MM < A(J,B + 3) THEN 2120
2130  IF J <  = I THEN 2150
2140  GOTO 2160
2150 A(I,B + 3) = MM:A(I,B + 2) = WW: GOTO 2220
2160 A(I,B + 3) = A(J,B + 3):A(I,B + 2) = A(J,B + 2):A(J,B + 3) =
    MM:A(J,B + 2) = WW:I = I + 1
2170  IF A(I,B + 3) < MM THEN 2190
2180  GOTO 2200
2190 I = I + 1: IF A(I,B + 3) < MM THEN 2190
2200  GOTO 2400
2210 A(J,B + 3) = A(I,B + 3):A(J,B + 2) = A(I,B + 2):J = J - 1: GOTO
    2100
2220 K = K + 1
2230  IF RR - I <  = I - LL THEN 2250
2240  GOTO 2260
2250 L(K) = I + 1:R(K) = RR:RR = I - 1: GOTO 2270
2260 L(K) = LL:R(K) = I - 1:LL = I + 1
2270  IF RR - LL >  = MM THEN 2090
2280  FOR J = LL + 1 TO RR
2281  PRINT ""
2290 BK = A(J,B + 3):FT = A(J,B + 2):I = J - 1
2300  IF BK < A(I,B + 3) AND I >  = LL THEN 2320
2310  GOTO 2330
2320 A(I + 1,B + 3) = A(I,B + 3):A(I + 1,B + 2) = A(I,B + 2):I =
    I - 1: IF BK < A(I,B + 3) AND I >  = LL THEN 2320
2330 A(I + 1,B + 3) = BK:A(I + 1,B + 2) = FT
2340  NEXT J
2350  IF K >  = 0 THEN 2060
2390  GOTO 2430
2400  IF J <  = I THEN 2420
2410  GOTO 2210
2420 I = J: GOTO 2220
2430  DIM XX(N - 1)
2431  FOR I = 0 TO N - 1
2432 XX(I) = A(I,B + 2)
2433  NEXT I
2434  FOR I = 0 TO N - 1
2435 ZZ = XX(I):A(ZZ - 1,B + 2) = N - I
2436  NEXT I
2500 DE = 1
2510  FOR I = N - 1 TO 0 STEP  - 1
2512  PRINT DE
2515  IF I = 0 THEN 5000
2520  IF A(I,B + 3) = A(I - 1,B + 3) THEN 2700
2522 DE = DE + PE + 1
2524 PE = 0
2530  NEXT I
2700  FOR J = 0 TO N - 1
2710  IF A(J,B + 2) = DE + PE + 1 THEN 2715
2712  GOTO 2720
2715 A(J,B + 2) = DE:PE = PE + 1
2716  GOTO 2530
2720  NEXT J
5000  FOR I = 0 TO N - 1
5010  FOR J = 0 TO B + 3
5020  PRINT A(I,J);"  ";
5030  NEXT J: PRINT
5040  NEXT I
```

4 apple

디스켓의 사용방법



4-1 디스켓의 사용방법

디스크 드라이브(disk drive)는 카세트 레코더보다 훨씬 빨리 사용하기 쉬운 것이나, 디스켓을 소중히 하도록 유의하지 않으면 안된다. 강한 자기장, 열, 구부림은 디스켓에 영향을 미친다. 그러므로 장치들을 자기장이나 열을 일으키는 텔레비전 수상기 같은 기구 가까이 놓아서는 안된다.

디스크 장치는 디스켓이 없거나 이니셜라이즈(initialize) 안 된 디스켓으로 작동되면 끝없이 작동을 계속하게 되며 손상을 입을 수도 있다. RESET 키를 누르면 멈추게 된다.

디스켓을 넣기 위해서는 디스크 장치의 뚜껑을 살짝 열고 디스켓의 표지를 위로하여 조심스럽게 밀어넣는다. 디스켓은 오른손을 사용하여 엄지손가락으로 표지 위를 잡고 검지 손가락은 아래를 잡아야 한다.

APPLE Ⅱ PLUS의 경우, 디스켓은 기계가 작동되면 자동적으로 사용가능상태로 된다. 애플에서 디스크를 인식하기 위해서는

PR#, IN#

```
PR#6        혹은        IN#6
```

을 타이프한다. 표준 컨트롤러 카드의 장소는 슬롯(slot) 번호 #6 이다. 다음에는 RETURN 키를 누른다.

프롬프트 기호가 *인 모니터에서 DOS를 부트하려면,

```
C600G
```

으로 하든가 6 CTRL-P 로 하면 된다.

INIT

새로운 디스켓을 사용하기 전에 이니셜라이즈되어야 한다. 우선 NEW를 타이프하고 디스켓을 명명하는 확인 프로그램을 작성한다.

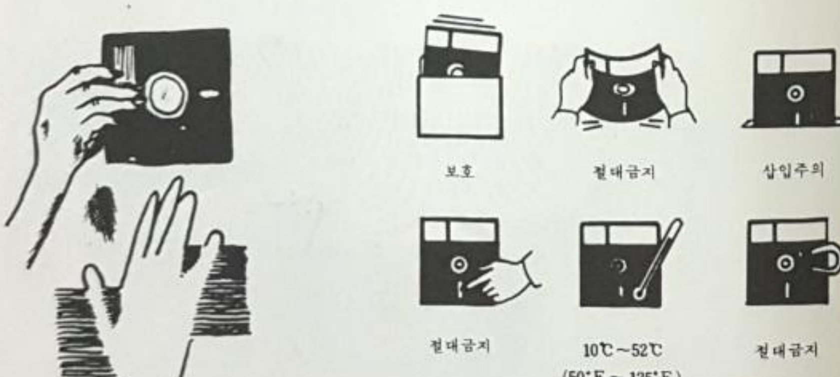


그림 4-1 디스켓 사용시 주의사항

```
* PROGRAM 4.1
 5 REM GREETING-1 PROGRAM
10 PRINT "SLAVE DISKETTE CREATED
     ON 48K SYSTEM"
20 PRINT "BY KWACK SUNG KEUN ON
     1 JANUARY 1984"
30 END
```

자기의 이름과 시스템의 크기, 날짜, 그것에 디스켓의 이력이나 슬레이브/마
스터의 구별이 빠르고, 간단하게 알 수 있는 정보를 넣어 두는 편이 편리하다. 기
대대로 되어 있는가 어떤가를 보기 위해서 이 프로그램을 RUN시켜 보자.

그리팅(greeting) 프로그램에 만족되면,

```
INIT HELLO
```

라고 타이프하고 **RETURN** 키를 누르면 디스켓은 약 2분간 회전하게 된다.
이니셜라이즈가 끝나면 프롬프트 기호가 표시되고 디스켓의 회전이 멈춘다. 확인
하기 위해 한번 더 드라이브로 되돌려 부트하면 PRINT문의 메세지가 표시된다.

```
SLAVE DISKETTE CREATED ON 48K SYSTEM
BY KWACK SUNG KEUN ON 1 JANUARY 1984
```

블랭크였던 디스켓이 이제 부트(BOOT)* 되었으므로 올바르게 이니셜라이즈
된 것을 알 수 있다.

*註 BOOT는 BOOT STRAP의 약칭으로 pull one self up by ones bootstrap이라는 표현이다.

MAXFILES

정상적으로 애플은 최고 3개의 화일을 동시에 개방할 수 있다. 이런 한계를 늘
리기 위해 직접방식으로 MAXFILES n을 타이프한다. 여기서 n은 일시에 개방할
수 있는 화일의 갯수를 말한다. 그러나 n이 커지면 커질수록 반대로 컴퓨터 기억
장치에서 당신의 프로그램에 쓸 수 있는 공간은 줄어들게 될 것이다.

개방되는 화일은 각각 기억장치에서 265바이트의 buffer를 차지한다.

```
MAXFILES 5
```

SAVE

디스크에 프로그램을 저장하려면, SAVE를 타이핑하고 그 다음에 화일명을 타
이프한다. 이는 다음에 디스크에 있는 프로그램을 확인하는데 사용된다.

```
SAVE STAR
```

이때 화일명의 길이는 30문자 이내가 아니면 안된다. DOS는 지나치게 긴 화일
명을 30문자로 끊어버린다. 화일명은 알파벳으로 시작하지 않으면 안된다. 코머
(,)를 제외하면 어떤문자라도 화일명으로 사용된다. 다음에 올바른 화일명을 표
시한다.

```
SOMNAMBULISTICS
ATOM 20
ELECTRONIC PHONE PAD:AAAAAAAA
```

다음에 사용할 수 없는 화일명과 그 이유를 표시한다.

```
1 TO 10                                   (숫자로 되어있다)
HIRES, HGR                                (코머가 들어있다)
INEPT EXCESS VERBIAGE DISQUALIFIES NAMES  (30문자 이상이다)
```

최후의 화일명은 CATALOG로 표시될 때에는 30문자로 잘라 줄여지나, LOAD나 RUN할 때에 타이프되면 전부의 명칭을 타이프하여도 정확하게 작용한다.

CATALOG

「STAR」 프로그램을 디스켓에 SAVE 하였다고 하면, 실제 디스켓상에는 다른 프로그램을 이미 SAVE하고 있었는지도 모른다. 디스켓에 어떤 프로그램이 들어 있는가를 보기 위해서는 다음의 명령을 타이프한다.

```
]CATALOG
```

하면 디스켓에 있는 모든 프로그램의 리스트가 표시된다. 다음 예를 보자.

```
DISK VOLUME 254

 A 002 HELLO
*A 052 SPACE INVADER
*A 093 ATOM 20
*B 067 AUTO BHAN
*A 060 ELECTRONIC PHONE PAD
 T 001 COUNTER
```

좌단의 「A」의 문자는 애플소프트, 「B」는 기계어, 그리고 「I」는 프로그램이 정수 BASIC이라는 것을 표시하고 있다. 또 「T」는 텍스트화일을 표시한다. 화일타이프를 표시하는 문자 다음의 숫자는 스토어한 프로그램의 길이를 표시한다. 여기서는 002디스켓 섹터가 프로그램을 스토어하는데 사용되고 있다. 각 섹터는 256바이트까지의 정보가 스토어된다. 하나의 화일이 255섹터를 넘으면 CATALOG로 표시되는 화일의 길이는 000에서 다시 한번 시작된다.

LOAD

자, 「STAR」라는 이름의 프로그램을 LOAD 시키려면,

```
LOAD STAR
```

의 명령을 사용한다. CATALOG 중에 이 프로그램명이 있으면, 그 프로그램이 LOAD 된다. 만일 이 프로그램명이 없다면,

```
FILE NOT FOUND
```

의 메세지를 낼 것이다. 디스켓의 CATALOG를 보고 프로그램의 올바른 화일명을 찾아내라. 문자와 스페이스 전부가 CATALOG 중의 화일명과 똑같도록 타이프하지 않으면 안된다.

DELETE

디스켓에 화일을 지우기 위해 이 명령어를 사용한다.

```
DELETE STAR
```

RENAME

어떤 이유로, 경우에 따라서는 화일명을 바꾸고 싶어질 때가 있다. 「MOON」이라는 화일을 「STAR」라는 명칭으로 바꾸고 싶으면 다음과 같이 타이프한다.

```
RENAME MOON, STAR
```

프로그램 중에서 DOS 명령을 실행시킬 경우는 매우 많고 극히 중요하다. 따라서 많은 DOS 명령이 BASIC 프로그램 중에서 실행된다. CTRL — D 의 다음에 코맨드를 계속한 스트링을 PRINT 하면 그 코맨드가 프로그램중에서 실행된다. 예로써

```
10 PRINT CHR$(4);"CATALOG"
```

등으로 사용할 수 있다.

4-2 시퀀설 화일 (Sequential File)

디스크 장치의 작동에 관계되는 모든 명령어는 **디스크 오퍼레이팅 시스템**(disk operating system; DOS)의 일부이다. 이들 명령어 중에 몇개는 직접방식으로 사용될 수도 있으며, 대부분이 프로그램에서도 사용될 수 있다. 프로그램에서 DOS명령어를 수행시키기 위해서는 PRINT문을 사용해야 한다. PRINT 다음에 CTRL — D 를 하고 그 다음에 명령어를 묶음표해야 한다. 예로써,

```
10 PRINT "CATALOG"
```

는 디스크 장치에 있는 디스켓의 화일과 프로그램의 **목록**(catalog)를 산출한다. CTRL-D는 CTRL 과 D 키를 동시에 누름으로써 컴퓨터에 입력되며 이것은 프린트되지 않는다.

CTRL 문자는 인쇄되는 것이 아니기 때문에 PRINT문 속에 CTRL 문자가 끼어 있는 프로그램은 때로는 읽기가 힘든다. 다음의 방법을 사용하면 CTRL-D의 존재를 확실히 할 수 있다.

```
* PROGRAM 4.2
  10 DOS$ = CHR$ (4): REM CTRL-D
  20 PRINT DOS$:"CATALOG"
```

CHR$(4)를 DOS$에 지정한 것은 DOS$를 CTRL-D와 같게 하는 역할을 한다. 컴퓨터는 PRINT문 처음에 CTRL-D를 만나면 다음에 오는 것은 무조건 DOS 명령어로 인식한다. 만일 그것이 올바른 DOS 명령어가 아닐 때는

```
?SYNTAX ERROR
]
```

라는 메세지가 인쇄되고 프로그램은 정지한다. PRINT문 처음에 CTRL-D가 없으면, DOS 명령어는 수행되지 않고 단순히 화면에 인쇄만 될 것이다.

다음 프로그램은 DOS 명령어가 CTRL-D가 있을 때와 없을 때는 어떻게 영향을 받는가를 보여준다.

```
* PROGRAM  4.3

10 DOS$ =  CHR$ (4)
20  PRINT "CATALOG": REM  NO CTRL
     -D
30  PRINT DOS$;"CATALOG"

]RUN
CATALOG


DISK VOLUME 254

 A 002 HELLO
*A 052 SPACE INVADER
*A 093 ATOM 20
*B 067 AUTO BHAN
*A 060 ELECTRONIC PHONE PAD
 T 001 COUNTER
*A 007 MAZE
 A 003 KWACK
 A 003 ANIMATED SQUARES
 A 003 WINKFACE
 A 011 KEYPICS
 A 006 ADD GR PAGES
 A 005 MUSIC
 A 003 P295
 A 002 E14
```

OPEN

OPEN 명령어는 컴퓨터에게 디스크에 있는 화일을 개방하여 사용할 준비를 하도록 지시한다.

```
* PROGRAM  4.4

10 DOS$ =  CHR$ (4)
20  PRINT DOS$;"OPEN MARKS"
```

는 컴퓨터한테 「MARKS」라는 이름의 화일을 찾도록 지시한다. 만일 그 화일이 없으면 새로운 이름이 화일목록에 가입되어, 후에 데이타가 그 화일에 저장될 수 있게 한다. 그러나 만일 「MARKS」라는 화일이 이미 있으면, 컴퓨터는 그 화일을 데이타의 원천으로 사용할 것이다.

프로그램과 화일은 둘 다 모두 디스크에 저장될 수 있기 때문에 각기 프로그램과 화일에는 다른 이름이 사용되어야 한다. 예로써 「MARKS」라는 이름의 화일을 악세스하는데 「AVERAGE」라는 이름의 프로그램이 사용될 수 있다. 이름을 중복으로 사용하는 것을 피하기 위해서는 당신은 모든 프로그램과 화일의 이름 기록을 관리하여야 한다.

CLOSE

일단 개방 open 된 모든 화일은 폐쇄(close)되야 한다. 화일을 폐쇄하기 위해 CLOSE 명령어를 사용하여 그 뒤에 화일명을 써 준다.

```
50  PRINT DOS$;"CLOSE MARKS"
```

는 「MARKS」 화일을 폐쇄한다. CLOSE문은 화일에 관한 모든 정보가 적절히 유지 보관되도록 하여 준다.

만일 프로그램이 CTRL—C 로 중지하거나 혹은 에러가 발생하여 화일이 개방된 채 중지한다면 직접방식으로 화일을 폐쇄시켜야 한다.

```
]PRINT CHR$ (4);"CLOSE MARKS"
```

OPEN문은 단순히 프로그램과 화일 사이의 연락선을 형성할 뿐이다. WRITE문은 당신으로 하여금 자료를 화일안에 넣도록 할 수 있게 한다.

```
30 DOS$ = CHR$ (4)
35 PRINT DOS$;"OPEN GRADES"
40 PRINT DOS$;"WRITE GRADES"
```

이면, 행번호 40이후에 나오는 모든 PRINT문은 그 화일이 폐쇄되거나 다른 DOS 명령어가 수행되기 전까지는 정보를 화면 대신에 화일로 보낸다.

다음의 프로그램은 「WORK」라는 이름의 **시퀀셜(順次) 화일**(sequential file)을 개방한다. 이 화일은 4명의 고용인의 이름과 시간당 임금, 노동시간을 수록한다.

```
# PROGRAM 4.5
10 DOS$ = CHR$ (4)
20 FOR X = 1 TO 4
30 INPUT "NAME? ";N$(X)
40 INPUT "WAGE? ";W(X)
50 INPUT "HOURS WORKED? ";H(X)
60 PRINT
70 NEXT X
80 PRINT DOS$;"OPEN WORK"
90 PRINT DOS$;"WRITE WORK"
100 FOR J = 1 TO 4
110 PRINT N$(J)
120 PRINT W(J)
130 PRINT H(J)
140 NEXT J
150 PRINT DOS$;"CLOSE WORK"
160 PRINT "THE INFORMATION IS IN
    THE FILE."
```

```
]RUN
NAME? HONG KIL DONG
WAGE? 5.02
HOURS WORKED? 37.12

NAME? KIM SA YUNG
WAGE? 4.45
HOURS WORKED? 39

NAME? LIM KKYUK JUNG
WAGE? 4.89
HOURS WORKED? 29.45

NAME? LEE MEANG DOL
WAGE? 4.76
HOURS WORKED? 41.4
THE INFORMATION IS IN THE FILE.
```

여기서 행번호 80과 90의 PRINT문은 화면에 나타나지 않는다. 이 두 PRINT문에서 처음에 나오는 CHR$(4)는 그 문장 나머지가 DOS 명령어라는 것을 가르쳐 준다. 행번호 110부터 130까지의 PRINT문의 내용도 화면에 나타나지 않는데 이것은 행번호 90에 있는 WRITE문이 컴퓨터에게 이후의 모든 PRINT문으로 부터의 데이타는 「WORK」라는 화일로 보내도록 지시했기 때문이다.

여기서 주시할 것은 행번호 90에 있는 WRITE 명령어 다음에 오는 PRINT문들은 화일에 정보를 보내 주지만 그것 자체가 DOS 명령어가 아니기 때문에 DOS$ 명령어를 포함하지 않는다는 것이다.

WRITE문을 실행하기 전에 우리는 화일에 수록할 정보를 모두 입력해야 한다. 만일 INPUT문이 WRITE문 이후에 실행되면 WRITE 명령어는 INPUT문으로부터 ?를 화면대신에 화일에 보내게 되어 화일에 있는 데이타를 파괴할 것이다. 이것을 방지하기 위해 화일에 넣을 모든 정보는 프로그램에서 WRITE 명령에 앞서 첨자변수에 저장하는 것이 좋다.

READ

READ 명령어 다음에 오는 INPUT문은 정보를 키보드 대신 화일로 부터 가져온다. 예로써, 만일 화일 「WORK」가 있을 때

```
* PROGRAM  4.6

  25 DOS$ =  CHR$ (4)
  30  PRINT DOS$;"OPEN WORK"
  35  PRINT DOS$;"READ WORK"
```

는 INPUT문이 나타날 때는 컴퓨터로 하여금 정보를 화일로 부터 가져오게 한다. 만일 「WORK」가 존재하지 않으면 컴퓨터는 키보드로 부터 데이타가 입력되기를 기다릴 것이다.

다음 프로그램은 이미 만들어진 화일 「WORK」를 개방하여서 4 명의 고용인에 대한 이름과 노동시간을 인쇄하도록 한다.

```
* PROGRAM  4.7

  10 DOS$ =  CHR$ (4)
  20  PRINT "NAME"; TAB( 25);
  30  PRINT "HOURS WORKED"
  40  PRINT DOS$;"OPEN WORK"
  50  PRINT DOS$;"READ WORK"
  60  FOR N = 1 TO 4
  70  INPUT N$
  80  INPUT W
  90  INPUT H
  100  PRINT N$; TAB( 30);
  110  PRINT H
  120  NEXT N
  130  PRINT DOS$;"CLOSE WORK"

]RUN
NAME                          HOURS WORKED
HONG KIL DONG                    37.12
KIM SA YUNG                      39
LIM KKYUK JUNG                   29.45
LEE MEANG DOL                    41.4
```

컴퓨터는 INPUT문에 의해 화일로 부터 가져올 바로 다음에 있는 정보를 가르키는 포인터를 가지고 있다. 하나의 화일이 개방될 때 그 포인터는 화일의 제일 앞에 위치한다. 즉 이 프로그램의 행번호 70에서 컴퓨터가 화일로 부터 읽는 첫번째 항목은 HONG KIL DONG으로 변수 N$에 대한 것이고, 다음은 변수 W에 대해 5.02를 읽고 그 다음은 변수 H에 대해 37.12를 읽는다. INPUT문이 나타날 때마다 포인터는 화일에서 데이타의 다음 항목을 가르키기 위해 달려간다.

행번호 80이 없었다면 5.02가 HONG KIL DONG의 노동시간으로 읽혀졌을 것이다. 다음번 루우프에서 37.12가 N$에 넣어지고 H에는 KIM SA YUNG이 넣어지게 된다. 후자의 작동과정은 컴퓨터가 에러 메세지를 발생하는 원인이 된다. 당신은 시퀀셜 화일의 구조와 그에 따른 INPUT문의 사용을 잘 기억하여야 한다. 프로그램에서 시간당 임금을 인쇄하지도 않으면서 행번호 80에서 그것을 입력한 것을 볼 수 있을 것이다. 이런 것은 컴퓨터가 화일을 올바른 순서로 읽기 위해서 필요한 것이다.

APPEND

이미 존재하는 시퀀셜 화일에서 이전에 수록된 정보에다 새로운 정보를 덧붙여 정보를 늘릴 수 있게 하는 명령이 APPEND이다.

```
* PROGRAM  4.8

  15 DOS$ =  CHR$ (4)
  20  PRINT DOS$;"APPEND WORK"
  30  PRINT DOS$;"WRITE WORK"
```

행번호 20에서 OPEN 명령어 대신 APPEND 명령어가 사용되었다. OPEN명령어가 실행될 때는 컴퓨터는 그 지정된 화일을 개방하고, 화일의 맨 앞부터 시작하여 화일에서 읽거나 쓸 준비를 한다. 만일 당신이 이전 화일에 정보를 덧붙인다면, 새로운 정보는 화일의 맨 앞에 이전 정보 위에다 쓰여지게 되고, 따라서 이렇게 되면 이전의 정보는 파괴될 것이다.

이처럼 지워지는 것을 방지하기 위해 APPEND 명령어는 이전의 화일을 개방하여 컴퓨터가 정보를 맨 뒤에다 쓰도록 준비한다.

다음의 프로그램은 이전의 화일 「WORK」를 추가하여 근래 채용된 2명의 새직원에 관한 정보를 포함시키려고 한다.

```
* PROGRAM  4.9
 10 DOS$ =  CHR$ (4)
 20  FOR Q = 1 TO 2
 30  INPUT "NAME? ";N$(Q)
 40  INPUT "WAGE? ";W(Q)
 50  INPUT "HOURS WORKED? ";H(Q)
 60  PRINT
 70  NEXT Q
 80  PRINT DOS$;"APPEND WORK"
 90  PRINT DOS$;"WRITE WORK"
 100  FOR T = 1 TO 2
 110  PRINT N$(T)
 120  PRINT W(T)
 130  PRINT H(T)
 140  NEXT T
 150  PRINT DOS$;"CLOSE WORK"
 160  PRINT "THE FILE HAS BEEN UPD
      ATED"

]RUN
NAME? LEE BU YUNG
WAGE? 4.67
HOURS WORKED? 43.26

NAME? PARK MIN SOO
WAGE? 4.90
HOURS WORKED? 39.18

THE FILE HAS BEEN UPDATED
```

APPEND 명령어는 항상 WRITE 명령어가 뒤따라야 한다. 만일 APPEND 뒤에 READ 명령어가 수행된다면 컴퓨터는 화일의 끝부분 뒤에서 데이타를 찾아보게 되므로 이런 과정은 에러를 유발시킨다.

단 한 문장으로 시퀀셜 화일에서 불필요한 정보를 삭제할 수는 없다. 하나의 화일을 변경시키려면 그것의 정보들을 새로운 화일로 옮겨야 한다. 필요한 정보만을 옮긴 후에 우리는 DELETE 명령어를 사용하여 이전의 화일을 지울 수 있다. 또 RENAME 명령어를 사용하여서 새로 만든 화일을 이전의 화일 이름으로 다시 이름지을 수 있다.

다음 프로그램은 직원이 퇴직할 때 화일 「WORK」로 부터 그 고용인의 명단을 삭제한다.

```
* PROGRAM  4.10
 10 DOS$ =  CHR$ (4)
 20  INPUT "WHO HAS RETIRED? ";R$
 30  PRINT DOS$;"OPEN WORK"
 40  PRINT DOS$;"OPEN TEMPORARY"
 50  FOR X = 1 TO 6
 60  PRINT DOS$;"READ WORK"
 70  INPUT N$
 80  INPUT W
 90  INPUT H
 100  IF N$ = R$ THEN 150
 110  PRINT DOS$;"WRITE TEMPORARY"
 120  PRINT N$
```

```
130 PRINT W
140 PRINT H
150 NEXT X
160 PRINT DOS$;"CLOSE WORK"
170 PRINT DOS$;"CLOSE TEMPORARY"

180 PRINT DOS$;"DELETE WORK"
190 PRINT DOS$;"RENAME TEMPORARY
    , WORK"
200 PRINT "THE INFORMATION ON ";
    R$
210 PRINT "HAS BEEN REMOVED."

]RUN
WHO HAS RETIRED? HONG KIL DONG
THE INFORMATION ON HONG KIL DONG
HAS BEEN REMOVED.
```

여기서 화일 「WORK」에 있는 모든 정보를 읽는다. 행번호 100은 퇴직한 고용인의 이름이 읽힐 때를 기다려서 그 사람을 빠뜨리고 넘어간다. 다른 모든 고용인에 관한 정보는 새로운 화일인 「TEMPORARY」에 놓인다. 이것이 완료되면 이전 화일 「WORK」는 행번호 180에서 파괴된다. 그리고 화일 「TEMPORARY」는 행번호 190에서 「WORK」라는 화일명으로 재 다시 주어진다. 새로운 화일 「WORK」는 퇴직한 사람에 관한 정보를 제외하고는 이 전의 것으로 부터의 모든 데이타를 수록하고 있다.

이 프로그램은 제한된 경우에서는 잘 수행되지만 이렇게 하는 방법은 항상 실질적이지는 못하다.

행번호 50과 150사이의 FOR … NEXT 루우프는 프로그램이 화일 끝의 뒤 쪽의 데이타를 읽지 못하도록 방지하고 있다. 이런 방법은 화일 「WORK」의 길이를 알기 때문에 사용되었다. 만일 화일의 길이를 모른다면 FOR … NEXT 루우프는 사용할 수 없다. 대신에 GOTO문이 화일을 전부 읽도록 사용되어야 한다. 그러면 컴퓨터가 화일의 끝을 만나기까지는 화일이 모두 읽혀지게 될 것이다.

이때

```
OUT OF DATA
```

의 메세지가 나타나면서 프로그램이 중지될 것이다. 이런 경우에, 화일은 개방된 채 남으며 중요한 데이타를 잃을 수도 있다. 에러가 발생했을 때 컴퓨터가 정지하는 것을 방지하기 위해 ONERR GOTO문이 사용된다.

ONERR GOTO문은 컴퓨터에게 에러가 발생했을 때 해야 할 일을 가르쳐 준다.

```
15 ONERR GOTO 375
```

만일 이 문장 이후에 에러가 발생하면 컴퓨터는 지정된 행번호로 간다. 프로그램에서 먼저 ONERR GOTO문을 두어서 에러가 발생하기 전에 컴퓨터가 그것을 알 수 있게 하는 것이 중요하다.

다음 프로그램은 앞의 프로그램을 수정하여 화일 「WORK」가 임의의 길이를 갖도록 허용한다.

여기서 화일에 없는 이름을 입력하였는가를 가르쳐 준다. 또한 ONERR GO-

* PROGRAM 4.11

```
5  ONERR  GOTO 160
10 DOS$ =  CHR$ (4)
15 T = 0
20  INPUT "WHO HAS RETIRED? ";R$
30  PRINT DOS$;"OPEN WORK"
40  PRINT DOS$;"OPEN TEMPORARY"
60  PRINT DOS$;"READ WORK"
70  INPUT N$
80  INPUT W
90  INPUT H
100  IF N$ = R$ THEN 155
110  PRINT DOS$;"WRITE TEMPORARY"

120  PRINT N$
130  PRINT W
140  PRINT H
150  GOTO 60
155 T = 1
158  GOTO 70
160  PRINT DOS$;"CLOSE WORK"
170  PRINT DOS$;"CLOSE TEMPORARY"

171  PRINT
172  IF T = 1 THEN 180
173  PRINT DOS$;"DELETE TEMPORARY
     "
174  PRINT "THERE IS NO ";R$
175  PRINT "ON THE PAYROLL"
176  END
180  PRINT DOS$;"DELETE WORK"
190  PRINT DOS$;"RENAME TEMPORARY
     , WORK"
210  PRINT "HAS BEEN REMOVED."
220  PRINT "OFF THE PAYROLL"

]RUN
WHO HAS RETIRED? KIM YUNG KI

THERE IS NO KIM YUNG KI
ON THE PAYROLL
```

TO와 GOTO 루우프를 사용하여 화일을 전부 읽는다. 여기서 변수 T는 찾을 이름이 화일에 있는지를 나타내는데 사용된다. 만일 있으면 T는 1이 되고, 없으면 0으로 된다. 어떤 시퀀셜 화일을 읽는데 이 방법이 가장 옳다.

일단 한번 에러에 걸렸으면 컴퓨터는 에러가 발생된 행번호로 보낼 것이다. 만일 조건부가 에러상태에서 바뀌지 않는다면, 프로그램은 원래 에러가 발생된 행번호에서 부터 수행을 계속하게 된다.

문제 135. 아래와 같은 5명의 국어, 영어, 수학 성적과 합계 점수를 디스크에 써넣는 프로그램을 작성하라.

A : 80, 96, 82　　B : 75, 82, 93　　C : 68, 97, 93

D : 88, 67, 98　　E : 78, 86, 88

```
10 DOS$ =  CHR$ (4)
20  PRINT DOS$;"OPEN SCORE"
30  PRINT DOS$;"WRITE SCORE"
40  FOR I = 1 TO 5
50  READ NAME$,KO,EN,MA
60 TT = 0
70 TT = KO + EN + MA
80  PRINT NAME$: PRINT KO: PRINT
     EN: PRINT MA: PRINT TT
90  NEXT I
100  PRINT DOS$;"CLOSE"
110  END
120  DATA  A,80,96,82
130  DATA  B,75,82,93
140  DATA  C,68.97,93
150  DATA  D,88,67,98
160  DATA  E,78,86,88
```

\#10～\#30
디스크 사용준비
화일명은 「SCORE」
\#40～\#80
5명의 이름, 성적과 합계데이타를 읽어 들여 화일에 써넣는다.
\#120～\#160
읽어들이는 데이타

> **문제** 136. 문제135 의 데이타에 추가하여 성명, 국어, 영어, 수학 점수를 입력하고 디스크에 저장하는 프로그램을 작성하라.

```
10 DOS$ = CHR$ (4)
20 INPUT "NAME? ";NA$
30 IF NA$ = "0" THEN 180
40 INPUT "KOREAN? ";KO
50 INPUT "ENGLISH? ";EN
60 INPUT "MATHEMATICS? ";MA
70 PRINT DOS$;"APPEND SCORE"
80 PRINT DOS$;"WRITE SCORE"
90 TT = 0
100 TT = KO + EN + MA
110  PRINT NA$
120  PRINT KO
130  PRINT EN
140  PRINT MA
150  PRINT TT
160  PRINT DOS$;"CLOSE"
170  GOTO 20
180  END
```

```
]RUN
NAME? F
KOREAN? 76
ENGLISH? 86
MATHEMATICS? 65
NAME? G
KOREAN? 87
ENGLISH? 64
MATHEMATICS? 46
NAME? 0
```

(NAME? F ~ ENGLISH? 64: 추가로 2사람의 성적기입)
(NAME? 0: ← 끝내기 위한 이름 "0"입력)

#20 - #30 : 이름이 "0"이면 끝난다.
#40 - #60 : 추가데이타 입력
#70 : 데이타의 추가이므로 open 대신 append를 이용하였다.
#80 : 화일「SCORE」에 써넣는다.
#100 : 성적합계
#110-#160 : 화일「SCORE」에 기록

4-3 랜덤화일(Random File)

랜덤화일(ramdom file)은 같은 크기의 레코드(record)들로 이루어 지는데, 레코드의 크기는 화일을 만들때 규정한다. 즉 랜덤화일은 서랍을 모아둔 것 같은 구조를 하고 있으며, 각 서랍은 독립적으로 사용될 수 있다.

이러한 보관 조직은 하나의 서랍모양과 같은 시퀀셜화일 보다는 확실히 우수하다. 시퀀셜화일에서는 데이타의 마지막 부분을 읽기 위해, 그 앞에 있는 모든 데이타를 차례대로 읽어야 한다. 그러나 랜덤화일에서는 데이타의 마지막 부분을 읽기 위해서 오직 마지막 서랍만 열면 된다. 이때 이 서랍을 **레코드**(record)라 한다. 이것은 정보를 저장하기 위해 예정된 공간으로 구성되어 있다. 레코드의 길이, 즉 가질 수 있는 문자의 갯수는 랜덤화일을 만들때 지정해 준다. 한 화일속에 있는 레코드들은 길수도 짧을 수도 있지만 그들은 모두 같은 길이를 가지고 있어야 한다.

한개의 레코드가 꼭 한개의 항목만 가질 필요는 없다. 총 문자의 수가 지정된 레코드의 길이를 넘지 않는 한도내에서 얼마든지 스트링이나 숫자를 가질 수 있다.

따라서 랜덤화일과 시퀀셜화일은 같은 명령을 사용하지만 다음과 같은 차이가 있다.

OPEN에서는 화일명, LP(P는 레코드의 길이)

READ 및 WRITE에서는 화일명, R#(#는 레코드의 번호)

자, 랜덤화일의 명령에 대해 알아 보자.

OPEN

랜덤화일은 OPEN문을 사용한다. 화일의 레코드의 길이는 길이표시인 L로 지정되어야 하며 그 다음에 레코드의 문자의 갯수가 뒤따라 온다.

다음은 화일의 각 레코드의 길이가 60자인 「KWACK」라는 이름의 랜덤화일을 생성한 것이다. 랜덤화일이 개방될 때마다 화일의 레코드의 길이를 지정하는 일이 중요하다. 그렇지 않으면 화일로 부터 읽거나 쓰는데 있어서 올바르게 되지않을 것이다.

```
20 PRINT DOS$;"OPEN KWACK,L60"
```

WRITE

랜덤화일로 사용될 때, WRITE명령은 정보를 화일의 지정된 레코드로 보낸다.

```
30 PRINT DOS$;"WRITE KWACK,R";A
```

는 화일 「KWACK」에 있는 레코드 번호A로 정보를 보낸다. 여기서 화일명 다음에 쉼표(,)와 문자 R(랜덤화일임을 표시)이 필요하다.

다음 프로그램은 화일 「ADDRESS」로서 이 화일에 5명의 이름과 주소를 넣을 수 있도록 하였다.

* PROGRAM 4.12

```
10 DOS$ = CHR$ (4)
20 PRINT DOS$;"OPEN ADDRESS,L30"

30 FOR X = 1 TO 5
40 INPUT "NAME? ";N$(X)
50 INPUT "DONG ADDRESS? ";A$(X)
60 PRINT
70 NEXT X
80 FOR Q = 1 TO 5
90 PRINT DOS$;"WRITE ADDRESS,R";
    Q
100 PRINT N$(Q)
110 PRINT A$(Q)
120 NEXT Q
130 PRINT DOS$;"CLOSE ADDRESS"
140 PRINT "THE INFORMATION IS IN
     THE FILE."
```

```
]RUN
NAME? HONG K.D.
DONG ADDRESS? 112 SINLIM-DONG

NAME? LEE G.S.
DONG ADDRESS? 167 HAGOK-DONG

NAME? PARK K.S.
DONG ADDRESS? 23 SINAM-DONG

NAME? KIM S.Y.
DONG ADDRESS? 435 BOOJYUN-DONG

NAME? CHOI S.H.
DONG ADDRESS? 231 SAMDUK-DONG

THE INFORMATION IS IN THE FILE.
```

여기서 행번호 80에서 변수Q는 각 이름과 주소가 들어갈 레코드의 번호를 지정한다.

각 레코드가 N$(Q)와 A$(Q)를 가지고 있는 랜덤화일은 다음과 같은 구조를 가지고 있다.

30

HONG K. D.	112 SINLIM-DONG
LEE G. S.	167 HAGOK-DONG
PARK K. S.	223 SINAM-DONG
KIM S. Y.	435 BOOJYUNG-DONG
CHOI S. H.	231 SAMDUK-DONG

각 레코드는 그 안에 있는 데이타가 30칸 이하로 차지할 때에도 그 길이는 30칸이다. 30자 이상을 이 프로그램에서 시도하면 결과적으로 30자 이후의 글자들은 없어진다. 빈칸도 하나의 문자로 취급한다. 물론 30자 이상으로 하고 싶으면 행번호 20의 L30을 그 이상으로 고치면 된다.

READ

랜덤화일은 우리에게 읽을 레코드의 위치를 알리도록 요구한다.

```
40  PRINT DOS$;"READ ADDRESS,R";A
```

는 컴퓨터에게 다음 INPUT문에서 A번 레코드를 읽도록 지시한다. 이렇게 특정한 레코드를 고를 수 있는 점에서 랜덤화일이 시퀸셜화일 보다 사용에 있어 효과적이다는 것이다.

즉 랜덤화일의 중요한 특징은 레코드 번호의 지정에 의해 화일내의 어느 곳에 위치한 데이타든지 바로 액세스(access)가 가능하다는 것이다.

다음 프로그램은 화일「ADDRESS」의 어느 레코드에 저장되어 있는 이름과 주소를 추출할 수 있도록 한 것이다.

```
* PROGRAM  4.13
5  ONERR  GOTO 90
10 DOS$ =  CHR$ (4)
12  PRINT "ENTER A NEGATIVE NUMBE
    R"
15  PRINT "TO STOP THE PROGRAM."
20  PRINT DOS$;"OPEN ADDRESS,L30"

30  INPUT "WHICH RECORD? ";I
35  IF I < 0 GOTO 110
40  PRINT DOS$;"READ ADDRESS,R";I

50  INPUT N$: INPUT A$
60  PRINT N$: PRINT A$
70  PRINT DOS$: PRINT
80  GOTO 30
90  PRINT "THE FILE IS NOT THAT L
    ONG."
100  GOTO 20
110  PRINT DOS$;"CLOSE ADDRESS"
```

```
]RUN
ENTER A NEGATIVE NUMBER
TO STOP THE PROGRAM.
WHICH RECORD? 1
HONG K.D.
112 SINLIM-DONG

WHICH RECORD? 3
PARK K.S.
23 SINAM-DONG

WHICH RECORD? 5
CHOI S.H.
231 SAMDUK-DONG

WHICH RECORD? -2
```

행번호 35에서 I<0을 입력하면, 즉 음수를 타이프하면 행번호 110에 의해 폐쇄될 것이다. 행번호 5의 에러처리문은 화일의 끝 부분이 넘는 레코드 번호가 지정되었을 경우에 당신에게 에러없이 프로그램을 정지하도록 알려주는 역할을 한다.

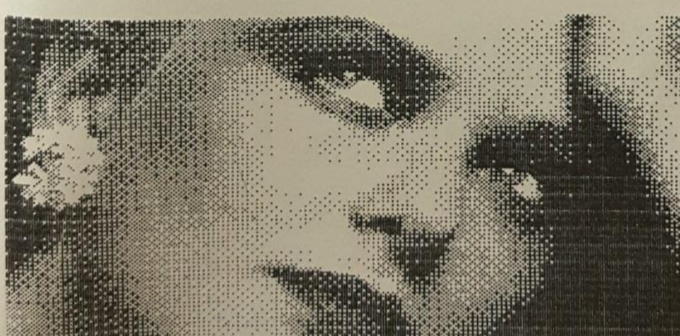
행번호 70에 있는 DOS $는 CTRL-D이므로 이것에 의해 행번호 40에서 수행된 READ문을 취소시키므로 행번호30의 INPUT은 데이타를 키보드로부터 요구하며, 화일로 부터 데이타를 사용하지는 않는다.

다음의 프로그램은 이전에 수록된 비트 이미지 형태와 정보인 LADY, COMPUTER TREE, APPLE LOGO, DOLLAR들을 화면에 읽어내는 프로그램의 예이다.

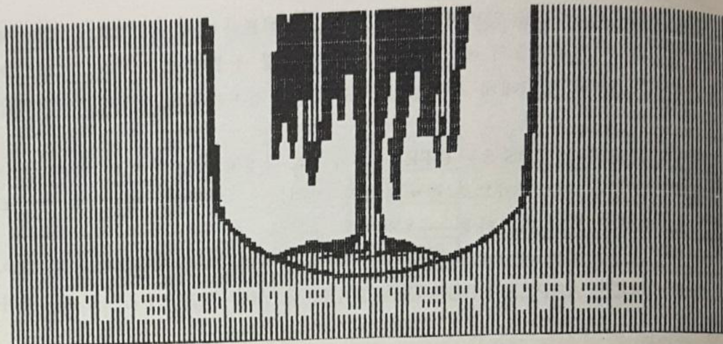
* PROGRAM 4.14

```
1  HGR : PRINT  CHR$ (4)"NOMONIOC
     ": HOME : GOTO 1000
3  POKE  - 16304, PEEK ( - 16297)
      OR  PEEK ( - 16301)
5  PRINT "BLOAD DROP IN/A$6000"
6  CALL 24784
7  POKE 24670,177: POKE 24671,19
8  POKE 24675,177: POKE 24676,19
9  CALL 24784
10  PRINT "BLOAD SHAKE/A$6000"
30  FOR I = 1 TO 60: CALL 24808
40  NEXT
50  FOR I = 1 TO 60: CALL 24880
70  NEXT
80  FOR I = 1 TO 20: CALL 24747
100  NEXT
110  FOR I = 1 TO 20: CALL 24688
130  NEXT
140  PRINT "BLOAD FLIP/A$300"
150  FOR I = 1 TO 10
160  FOR J = 1 TO 200: NEXT J: CALL
     880
190  NEXT
200  PRINT "BLOAD SEPERATE/A$6000
     "
210  FOR I = 1 TO 80: CALL 24808
230  NEXT : FOR I = 1 TO 80: CALL
     24880
260  NEXT : RETURN
1000  READ A$: IF  NOT  LEN (A$) THEN
      RESTORE : GOTO 1000
1010  PRINT  CHR$ (4)"BLOAD"A$",A
     $4000": GOSUB 3: IF  FRE (0)
      THEN 1000
2000  DATA  LADY,COMPUTER TREE,AP
     PLE LOGO,DOLLAR,
```

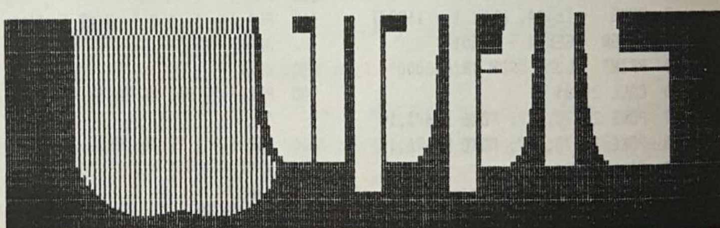
註
이 그림은 VTR 카메라에 의해 비트이미지 형태화된 디스켓을 이용하여 출력한 것임.



LADY



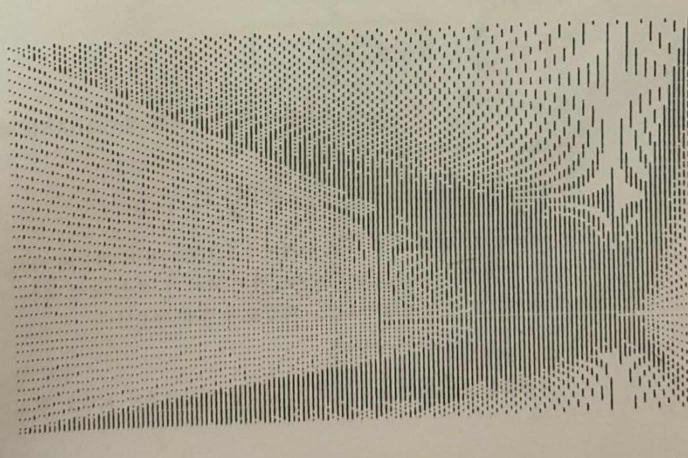
COMPUTER TREE



APPLE LOGO

5 apple

응용 및 게임(Game)



5-1 만년 캘린더(calender) 작성

캘린더(calender)를 작성하자면 캘린더가 어떤 규칙아래서 만들어졌느냐를 알아야만 한다. 우리가 말하는 서기는 「글레고리오曆」이라는 규칙에 따른 것이다.

1년을 365일로 한 것은 기원원년 무렵이며 이 때에는 4년에 한번씩 윤년을 마련하였다. 이것을 율리우스曆이라 일컫는다. 그런데 이 방식은 실제의 1년과는 오차가 생긴다. 128년이 경과하면 1일의 차이가 난다. 그래서 다시 정확한 캘린더를 만들기 위해 글레고리오 13세가 작성한 것이 글레고리오曆이다. 이것은 기원년수가 4로 완전히 나누어지는 해는 윤년으로 하지만 100의 배수일 때에는 400으로 나누어지는 해 이외는 평년으로 한다는 것이다. 이 曆法은 1만년에 3일의 차가 생기므로 일상생활에서는 무시해도 무방한 것이다. 이 역법은 1582년 10월부터 시행되었다. 그러므로 글레고리오曆으로 계산할 수 있는 것은 1583년 이후이고, 그 이전은 율리우스曆으로 계산하여야 한다.

여기서는 1583년 이후로 하여 글레고리오曆만을 생각하기로 하자.

캘린더를 작성하는 프로그램을 짜는데 있어서 문제점이 3가지가 있다.

1. 요일의 계산

주어진 년월에서 그달의 첫날이 무슨 요일인가를 계산하여야 한다. 우선 특정일의 요일을 데이타로 주어놓고 캘린더를 작성하는 달의 첫날과 그날의 차를내어 그 일수에서 요일을 계산하는 방법과 공식에 의해서 요일을 계산하는 방법이다. 여기서 후자의 방법으로 「**Zeller의 공식**」을 이용하자.

Zeller의 공식은 서기 N년M월D일이 무슨 요일인가를 계산하는 것이다. 여기서 글레고리오曆만을 사용하기 위해

$$1583 \leqq N \leqq 5000$$

로 해둔다.

$$C = [\frac{N}{100}], \quad Y = N - 100C$$

로 한다. $[\frac{N}{100}]$은 $\frac{N}{100}$를 초과하지 않는 최대의 정수, BASIC에서는 INT함수가 된다. 따라서 Zeller의 공식은

$$W = [\frac{C}{4}] - 2C + Y + [\frac{Y}{4}] + [\frac{26(M+1)}{26}] + D - 1$$

로 주어진다. W를 7로 나눈 나머지가 0,1,2,…6이 되는데 따라 구하는 날은 일요일, 월요일, …, 토요일이 된다. 다만 M이 1 또는 2일 때는 즉, 1월, 2월의 경우에는 각각 전년의 13월, 14월로서 W를 계산한다.

2. 달의 일수

다음달의 초하루의 요일을 W2, 그 달의 초하루의 요일을 W1으로 한다. 이

때 W1, W2는 0,1,2,…6이 된다.

W2 ≧ W1 이면 28+W2−W1

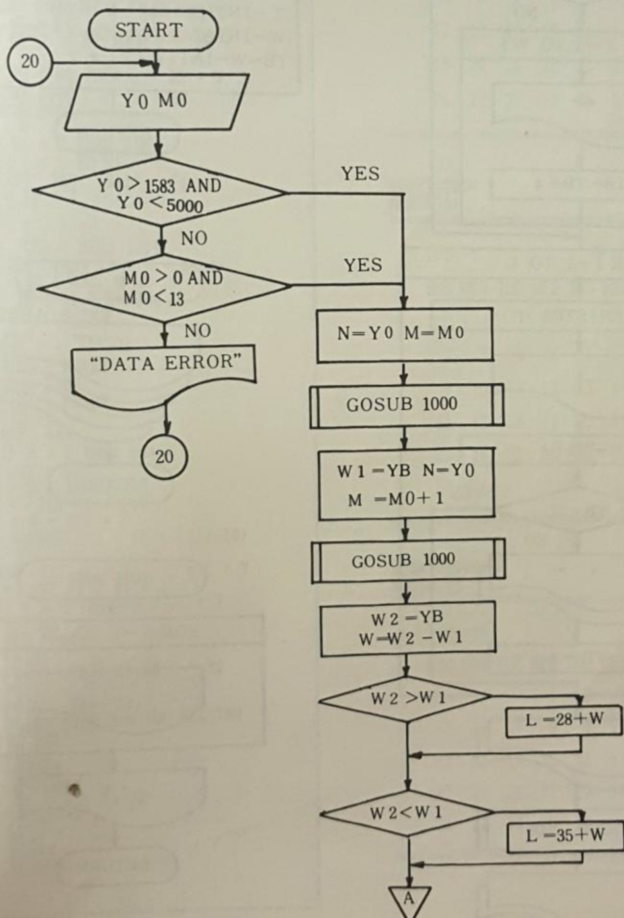
W2 < W1 이면 35+W2−W1

윗식이 그달의 일수가 된다.

3. 출력의 방법

1행에 1주일분을 치고 첫째 주는 도중까지 스페이스를 남기지 않으면 안된다.

그럼 만년 캘린더를 작성하기 위한 프로그램의 흐름도(FLOW CHART)를 그려보자.



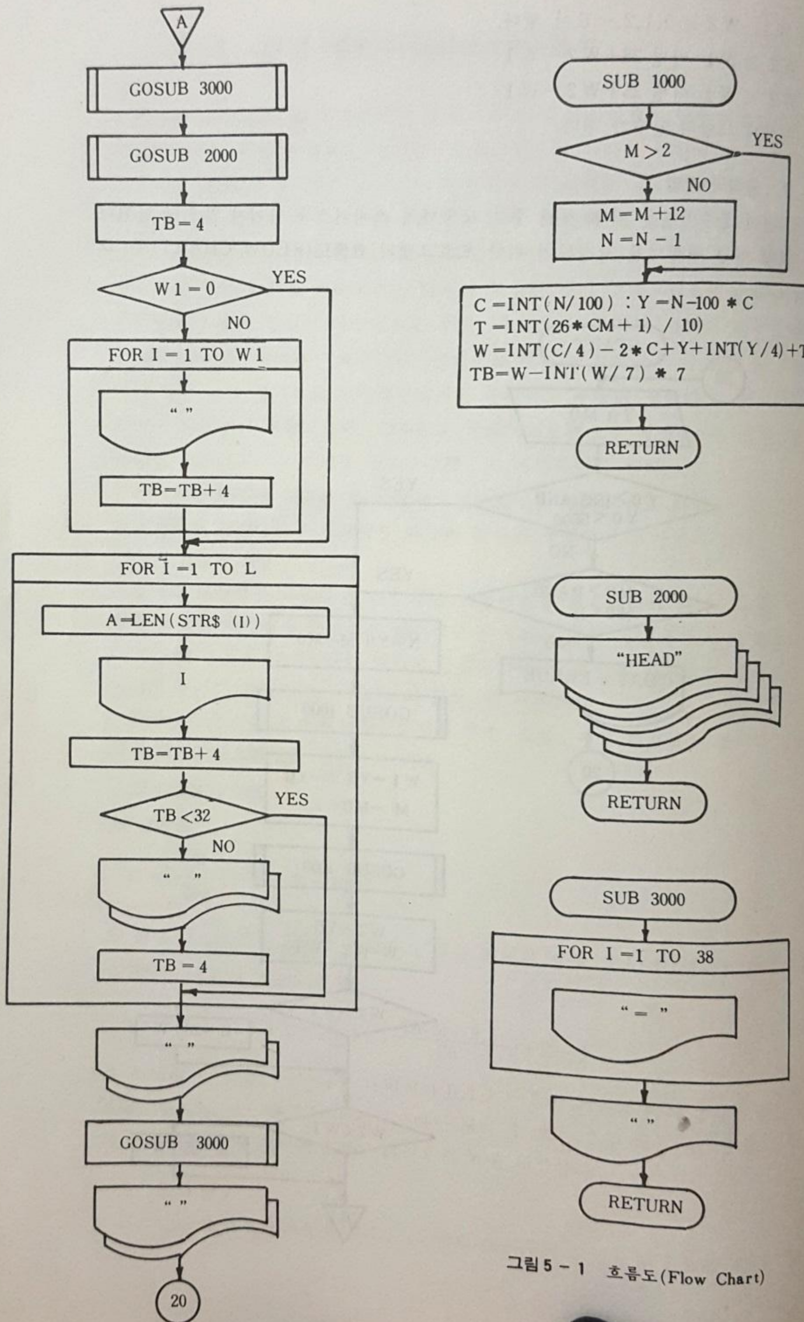


그림 5 - 1 흐름도(Flow Chart)

```
10  REM
11  REM  *********************
12  REM  * C A L E N D A R *
13  REM  *********************
14  REM
20  INPUT "YEAR?=";YO
30  INPUT "MONTH?=";MO
120  IF YO > 1583 AND YO < 5000 THEN
    160
130  IF MO > 0 AND MO < 13 THEN 1
    60
140  PRINT "DATA ERROR,REENTER"
150  GOTO 20
160 N = YO:M = MO
170  GOSUB 1000:W1 = YB
180 N = YO:M = MO + 1
190  GOSUB 1000:W2 = YB
200 W = W2 - W1
210  IF W2 >  = W1 THEN L = 28 +
    W
220  IF W2 < W1 THEN L = 35 + W
230  GOSUB 3000
240  GOSUB 2000
250 TB = 4
260  IF W1 = 0 THEN 310
270  FOR I = 1 TO W1
280  PRINT  TAB( TB);"   ";
290 TB = TB + 4
300  NEXT I
310  FOR I = 1 TO L
320 A =  LEN ( STR$ (I))
330  PRINT  TAB( TB + 4 - A);I;
340 TB = TB + 4
350  IF TB < 32 THEN 380
360  PRINT : PRINT
370 TB = 4
380  NEXT I
390  PRINT : PRINT
400  GOSUB 3000
410  PRINT : PRINT
420  GOTO 20
1000  REM  ***
1010  IF M > 2 THEN 1040
1020 M = M + 12
1030 N = N - 1
1040 C =  INT (N / 100)
1050 Y = N - 100 * C
1060 T =  INT (26 * (M + 1) / 10)

1070 W =  INT (C / 4) - 2 * C + Y
     +  INT (Y / 4) + T
1080 YB = W -  INT (W / 7) * 7
1090  RETURN
2000  REM  ***
2020  PRINT
2030  PRINT  TAB( 5);YO;"-";MO
2040  PRINT
2050  PRINT  TAB( 5);
2060  PRINT "SUM MON TUE WED THU
    FRI SAT"
2070  PRINT
2080  RETURN
3000  REM  ***
3010  FOR I = 1 TO 38
3020  PRINT "=";
3030  NEXT I
3040  PRINT
3050  RETURN
```

```
]RUN
YEAR?=1984
MONTH?=8
======================================

    1984-8

    SUM MON TUE WED THU FRI SAT

                     1   2   3   4

     5   6   7   8   9  10  11

    12  13  14  15  16  17  18

    19  20  21  22  23  24  25

    26  27  28  29  30  31

======================================

YEAR?=2000
MONTH?=1
======================================

    2000-1

    SUM MON TUE WED THU FRI SAT

                                 1

     2   3   4   5   6   7   8

     9  10  11  12  13  14  15

    16  17  18  19  20  21  22

    23  24  25  26  27  28  29

    30  31

======================================

YEAR?=1999
MONTH?=9
======================================

    1999-9

    SUM MON TUE WED THU FRI SAT

                     1   2   3   4

     5   6   7   8   9  10  11

    12  13  14  15  16  17  18

    19  20  21  22  23  24  25

    26  27  28  29  30

======================================
```

5-2 유아교육용 키보드픽쳐(Key board picture)

먼저 프로그램을 입력시킨 후 원하는 키를 누른다. 원하는 글자의 키를 누르면 그 글자가 텍스트부분의 첫칸에 보통 크기의 문자가 깜박거리며 그와 동일한 종류의 문자가 저해상도 그래픽모드의 블록으로 이루어진 도형문자로 변화되어 화면에 나타난다.

표시될 수 있는 문자는 알파벳 26자와 아라비아숫자 10개 그리고 키보드에 있는 모든 종류의 특수문자 예로써 @, *, #, ? 등이다.

보통의 녹색 및 회색 모니터로 이 프로그램을 동작시키면 문자의 색은 동일하나 실제로 컬러모니터를 사용할 경우, 표현되는 문자의 색은 여러가지로 나타난다.

전체프로그램은 약 2.7K 바이트 정도이고,행번호 10~180은 주 프로그램이며 행번호 1000~1094는 각 문자의 형태를 규정하는 데이타이다.

행번호 50은 화면에 나타내고 싶은 문자를 한자씩 읽어 들여 화면의 위에서부터 21번째 줄에 프린트하고 행번호 70에서는 화면에 나타낼 문자의 색을 선택한다. 0에서 15사이의 난수로 16가지의 색을 선택할 수 있게 된다. 행번호 90은 행번호 50에서 읽어들인 문자를 ASCII코드의 값으로 바꾸어 33을 빼준다. 왜냐하면 ASCII코드로 0에서 32사이에는 특별한 문자가 지정되어 있지 않기 때문이다.

행번호 100과 110에서 흔히 표현할 수 있는 문자의 범위를 정하고 있다. 만일 알파벳 소문자나 한글자모를 표현하고 싶으면 행번호 90, 100, 110, 1020을 적절히 고치고 새로운 문자를 정의해야 된다. 행번호 130~160의 프로그램으로 원하는 문자를 그리게 된다. 여기서 문자를 그리기 위해서는 DATA문을 먼저 볼 필요가 있다.

행번호 1005에서 B$의 값을 읽어들인다. 이것은 화면에 그릴 문자를 2진수로 나타낸 후 어느 정도의 길이로 문자를 그릴 것인지 결정하게 된다. 행번호1020에서 LI$에 해당하는 문자를 읽고 행번호 1026에서 LI$에 해당되는 행을 읽어들여 행번호 1005에서 읽어들인 B$의 값으로 대치한다.

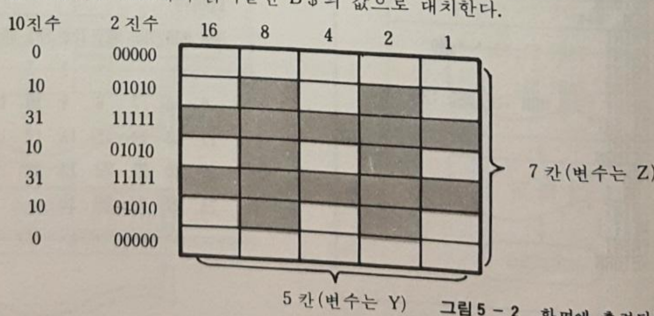


그림 5-2 화면에 출력되는 #기호

예를들어 키보드에서 #를 눌렀다고 하면 #의 ASCII코드가 35이므로 X의 값은 2가 된다. 행번호 1022에서 읽어들인 LI$의 값이 #와 같고 행번호 1035의 데이타에서 #의 값에 해당하는 B의 값은 0, 10, 31, 10, 31, 10, 0이며, L$의 값은 00000, 01010, 11111, 01010, 11111, 01010, 00000으로 변환된다.

따라서 화면에는 그림5-2와 같은 문자가 나타난다. L$의 값은 일종의 비트패턴과 동일한 개념의 출력방식이다.

```
1  REM ***********************
2  REM *  KEY BOARD PICTURE  *
3  REM ***********************
4  REM
10  DIM LI$(61),L$(61,7),B$(31)
20  HOME
30  GOSUB 1000
40  GR
50  VTAB 21: GET A$: PRINT A$
60  GR : REM  TO ERASE LAST LETTE
    R
70 C =  INT ( RND (1) * 16): IF C
     = 0 THEN 70
80  COLOR= C
90 X =  ASC (A$) - 33
100  IF X < 0 THEN 50
110  IF X > 62 THEN 50
120 R =  INT ( RND (1) * 35):S =
     INT ( RND (1) * 33)
130  FOR Z = 1 TO 7
140  FOR Y = 1 TO 5
150  IF  MID$ (L$(X,Z),Y,1) = "1"
     THEN  PLOT Y + R,Z + S
160  NEXT : NEXT
170  GOTO 50
180  END
1000  REM  BINARY DATA
1005  FOR I = 1 TO 31: READ B$(I)
     : NEXT
1010  DATA  00001,00010,00011,001
     00
1011  DATA  00101,00110,00111,010
     00
1012  DATA  01001,01010,01011,011
     00
1013  DATA  01101,01110,01111,100
     00
1014  DATA  10001,10010,10011,101
     00
```

```
1015  DATA  10101,10110,10111,110
     00
1016  DATA  11001,11010,11011,111
     00
1017  DATA  11101,11110,11111
1020  FOR X = 0 TO 61
1022  READ LI$(X)
1024  FOR Z = 1 TO 7
1026  READ B:L$(X,Z) = B$(B)
1028  NEXT : NEXT
1030  RETURN
1031  REM  PICTURE DATA
1033  DATA "!",4,4,4,4,4,0,4
1034  DATA "'",10,10,10,0,0,0,0
1035  DATA "#",0,10,31,10,31,10,0
1036  DATA "$",4,15,20,14,5,30,4
1037  DATA  "%",25,26,4,4,8,11,19
1038  DATA  "&",12,20,8,12,21,18,
     13
1039  DATA "'",12,12,4,0,0,0,0
1040  DATA "(",2,6,4,8,4,6,2
1041  DATA ")",8,12,4,2,4,12,8
1042  DATA "*",4,21,14,31,14,21,4
1043  DATA "+",0,4,4,31,4,4,0
1044  DATA ",",0,0,0,0,6,6,2
1045  DATA "-",0,0,0,31,0,0,0
1046  DATA ".",0,0,0,0,6,6,0
1047  DATA "/",0,1,2,4,8,16,0
1048  DATA  0,14,10,17,17,17,10,1
     4
1049  DATA  1,4,12,4,4,4,4,14
1050  DATA  2,14,17,19,2,4,8,31
1051  DATA  3,14,19,1,14,1,19,14
1052  DATA  4,6,10,18,31,2,2,2
1053  DATA  5,31,16,16,30,3,17,14
1054  DATA  6,14,25,16,22,25,17,1
     4
1055  DATA  7,31,17,2,4,4,8,8
```

```
1056  DATA  8,14,17,10,4,10,17,14
1057  DATA  9,14,17,19,13,1,19,14
1058  DATA ":",0,12,12,0,12,12,0
1059  DATA ";",0,12,12,0,12,12,4
1060  DATA "<",0,2,4,8,4,2,0
1061  DATA "=",0,0,31,0,31,0,0
1062  DATA ">",0,8,4,2,4,8,0
1063  DATA "?",14,17,2,4,4,0,4
1064  DATA "@",14,17,21,23,18,16,
      15
1065  DATA  A,4,14,10,14,27,17,17

1066  DATA  B,28,18,22,28,22,18,2
      8
1067  DATA  C,14,25,16,16,16,25,1
      4
1068  DATA  D,28,18,17,17,17,18,2
      8
1069  DATA  E,31,16,16,30,16,16,3
      1
1070  DATA  F,31,16,16,30,16,16,1
      6
1071  DATA  G,14,25,16,16,23,18,1
      4
1072  DATA  H,17,17,17,31,17,17,1
      7
1073  DATA  I,14,4,4,4,4,4,14
1074  DATA  J,2,2,2,2,2,18,12
1075  DATA  K,17,18,20,24,20,18,1
      7
1076  DATA  L,16,16,16,16,16,16,3
      0
```

```
1077  DATA  M,27,27,21,21,17,17,1
      7
1078  DATA  N,17,25,25,21,19,19,1
      7
1079  DATA  O,14,27,17,17,17,27,1
      4
1080  DATA  P,30,19,17,19,28,16,1
      6
1081  DATA  Q,14,27,17,17,21,18,1
      3
1082  DATA  R,30,19,17,19,28,22,1
      7
1083  DATA  S,14,17,24,4,3,17,14
1084  DATA  T,31,4,4,4,4,4,4
1085  DATA  U,17,17,17,17,17,27,1
      4
1086  DATA  V,17,17,27,10,10,14,4

1087  DATA  W,17,17,17,21,21,27,1
      0
1088  DATA  X,17,27,10,4,10,27,17

1089  DATA  Y,17,10,10,4,4,4,4
1090  DATA  Z,31,1,2,4,8,16,31
1091  DATA  0,0,0,0,0,0,0,0
1092  DATA  0,0,0,0,0,0,0,0
1093  DATA  0,0,0,0,0,0,0,0
1094  DATA "^",4,10,17,0,0,0,0
2000  REM  LAST 2 DIGITS OF LINES
      1033 TO 1094 ARE ASCII CODE
      OF CHARACTER
```

5-3 애플컴퓨터(Apple Ⅱ Computer) 모형

이 프로그램은 매우 긴 편이나 구성은 간단하다. 프로그램이 긴 이유는 개별적으로 정의되어야 할 점이 매우 많기 때문이다.

프로그램을 작성할 때, 물체의 좌측 하단 후미의 구석점을 기준점으로 하여 여기서부터 인치단위의 근사값들이 구하여져 사용되었다. 일부 측정값은 DATA문으로 주어지는데, 이것들은 각각 그림을 그리는데 사용되는 점들의 X, Y, Z 좌표값을 나타낸다. 또 이 프로그램 내에서는 화면에 나타내어질 점들의 화면 좌표들을 얻기 위해 행번호 20에서 60까지의 투영서브루틴이 사용되었다.

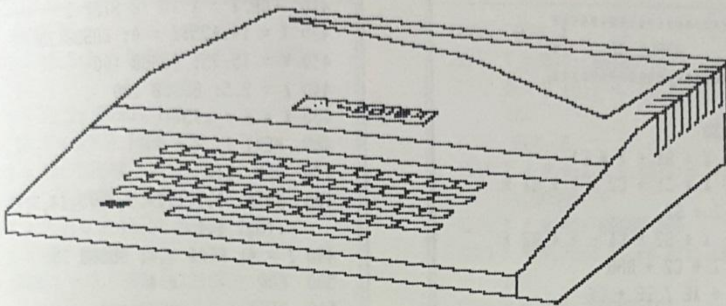


그림 5 - 3

행번호 1490에서 1600까지에서는 **흑색지움**(blackout) 방법을 이용하여 은선제거를 행하고 있다.

변수T는 두가지 목적으로 사용되는데, 첫째는 흑색지움루틴에서 사용되는 화면의 상단을 나타내는 Y좌표를 나타내는데 사용되었다. 즉 첫째 화상에 대해 T=0, 둘째 화상에 대해서는 T=40이다. 두번째로는 T가 둘째 화상이 그려질 때 프로그램의 적당한 부분으로 이동하는데 사용되었다. 결과로서 두번째의 작은 화상은 첫번째 화상만큼 세밀하지 못하다. 행번호 1940과 1950에 나타나듯이 세번째 화상은 단지 두개의 사각형으로 간단하게 그려진다.

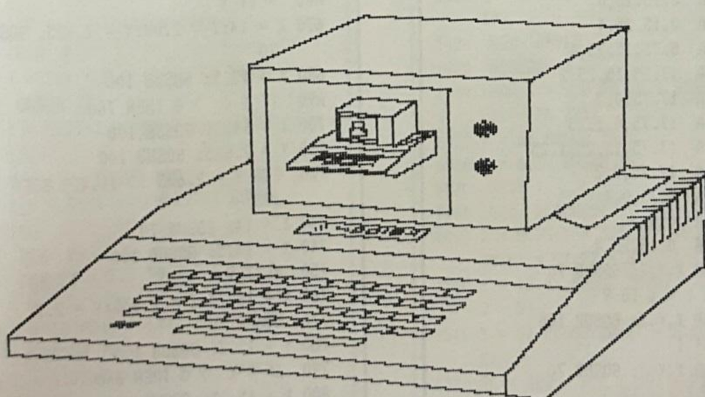


그림 5 - 4

```
1  REM *********************
2  REM *      APPLE ][     *
3  REM *********************
4  REM
10  GOTO 130
20 XE =  - X * S1 + Y * C1
30 YE =  - X * C1 * C2 - Y * S1 *
    C2 + Z * S2
40 ZE =  - X * S2 * C1 - Y * S2 *
    S1 - Z * C2 + RHO
50 SX = D * XE / ZE + CX
60 SY = CY - D * YE / ZE: RETURN
70  GOSUB 20
80  HPLOT SX,SY
90  RETURN
100  GOSUB 20
110  HPLOT  TO SX,SY
120  RETURN
130 RHO = 100:D = 1100:CX = 116:C
    Y = 105
140 THETA = .45:S1 =  SIN (TH):C1
     =  COS (TH)
150 PHI = 1.25:S2 =  SIN (PH):C2 =
     COS (PH)
160  HGR2 : HCOLOR= 3
170  DATA   17.75,15.25,2
180  DATA   17.75,15.25,1.25
190  DATA  15,15.25,0
200  DATA  0,15.25,0
210  DATA  0,15.25,4
220  DATA  8.75,15.25,4
230  DATA  17.75,15.25,2
240  DATA  17.75,0,2
250  DATA  17.75,0,1.25
260  DATA  17.75,15.25,1.25
270  DATA  17.75,0,2
280  DATA  8.75,0,4
290  DATA  0,0,4
300  DATA  0,15.25,4
310  READ X,Y,Z: GOSUB 70
320  FOR I = 1 TO 9
330  READ X,Y,Z: GOSUB 100
340  NEXT I
350  READ X,Y,Z: GOSUB 70
360  FOR I = 1 TO 3
370  READ X,Y,Z: GOSUB 100
380  NEXT I
390  IF T <  > 0 THEN 470
400 X = .875
410  FOR I = 1 TO 18 STEP 2
420 Y = 14.625:Z = 4: GOSUB 70
430 Y = 15.25: GOSUB 100
440 Z = 2.5: GOSUB 100
450 X = X + .75
460  NEXT I
470  REM  LID
480  DATA  1,1,1,14.3,7.75,14,3,7
    .75,1,1,1
490 Z = 4: READ X,Y: GOSUB 70
500  FOR I = 1 TO 4
510  READ X,Y: GOSUB 100
520  NEXT I
530 Y = 0:X = 11:Z = 3.5: GOSUB 7
    0
540 Y = 15.25: GOSUB 100
550 X = 12.25:Z = 3.22: GOSUB 70
560 Y = 0: GOSUB 100
570  REM  KEYBOARD
580 X = 13.25:Z = 3.125:Y = 3: GOSUB
     70
590 Y = 12.5: GOSUB 100
600  IF T <  > 0 THEN 670
610 X = 13.75: GOSUB 100
620 Y = 3: GOSUB 100
630  FOR Y = 3 TO 11.77 STEP .73
640 X = 13.25: GOSUB 70
650 X = 13.75: GOSUB 100
660  NEXT Y
670 X = 14:Z = 2.88:Y = 2.625: GOSUB
     70
680 Y = 12.5: GOSUB 100
690  IF T <  > 0 THEN 760
700 X = 14.5: GOSUB 100
710 Y = 2.625: GOSUB 100
720  FOR Y = 2.625 TO 11.425 STEP
    .73
730 X = 14: GOSUB 70
740 X = 14.5: GOSUB 100
750  NEXT Y
760 X = 14.75:Z = 2.635:Y = 2.7
770  GOSUB 70
780 Y = 12.2: GOSUB 100
790  IF T <  > 0 THEN 860
800 X = 15.25: GOSUB 100
810 Y = 2.7: GOSUB 100
820  FOR Y = 2.7 TO 12.2 STEP .73
830 X = 14.75: GOSUB 70
840 X = 15.25: GOSUB 100
```

```
850  NEXT Y
860 X = 15.5:Z = 2.39:Y = 2.7: GOSUB
     70
870 Y = 12.2: GOSUB 100
880  IF T < > 0 THEN 970
890 X = 16: GOSUB 100
900 Y = 2.7: GOSUB 100
910 Y = 2.7:X = 15.5: GOSUB 70
920 X = 16: GOSUB 100
930  FOR Y = 3.7 TO 11.2 STEP .73

940 X = 15.5: GOSUB 70
950 X = 16: GOSUB 100
960  NEXT Y
970 X = 16.25:Z = 2.145:Y = 4.7: GOSUB
     70
980 Y = 10.45: GOSUB 100
990  IF T < > 0 THEN 1070
1000 X = 16.75: GOSUB 100
1010 Y = 4.7: GOSUB 100
1020 X = 16.25: GOSUB 100
1030  FOR Y = 2.7 TO 3.45 STEP .1

1040 X = 16.25: GOSUB 70
1050 X = 16.75: GOSUB 100
1060  NEXT Y
1070  DEF  FN Z(X) = 5.94 - .2 *
     X
1080  IF T < > 0 THEN 1480
1090  REM  A
1100 Y = 7:X = 9.95:Z =  FN Z(X):
     GOSUB 70
1110 X = 9.75:Z =  FN Z(X): GOSUB
     100
1120 Y = 6.7: GOSUB 100
1130 X = 9.95:Z =  FN Z(X): GOSUB
     100
1140  REM  PP
1150  FOR LY = 7.2 TO 7.7 STEP .5

1160 X = 10.05:Y = LY:Z =  FN Z(X
     ): GOSUB 70
1170 X = 9.75:Z =  FN Z(X): GOSUB
     100
1180 Y = LY + .3: GOSUB 100
1190 X = 9.95:Z =  FN Z(X): GOSUB
     100
1200 Y = LY: GOSUB 100
1210  NEXT LY
1220  REM  L
1230 Y = 8.2: GOSUB 70
1240 X = 9.65:Z =  FN Z(X): GOSUB
     100
1250  REM  E
1260 Y = 8.8:X = 9.95:Z =  FN Z(X
     ): GOSUB 70
1270 Y = 8.45: GOSUB 100
1280 X = 9.75:Z =  FN Z(X): GOSUB
     100
1290 Y = 8.8: GOSUB 100
1300 X = 9.95:Y = 8.42:Z =  FN Z(
     X): GOSUB 100
1310  REM  RIGHT BRACKET
1320 Y = 9: GOSUB 70
1330 X = 9.65:Z =  FN Z(X): GOSUB
     100
1340 Y = 9.25: GOSUB 70
1350 X = 9.95:Z =  FN Z(X): GOSUB
     100
1360  REM  BOX
1370 Y = 5.5:X = 10.4:Z =  FN Z(X
     ): GOSUB 70
1380 Y = 9.5: GOSUB 100
1390 X = 9.4:Z =  FN Z(X): GOSUB
     100
1400 Y = 5.5: GOSUB 100
1410 X = 10.4:Z =  FN Z(X): GOSUB
     100
1420  REM  APPLE
1430  HCOLOR= 5
1440  FOR X = 9.8 TO 10.1 STEP .1
     :Z =  FN Z(X)
1450 Y = 6: GOSUB 70
1460 Y = 6.3: GOSUB 100
1470  NEXT X
1480  REM  BLACKOUT
1490 X = 8.75:Y = 3.4:Z = 4: GOSUB
     20:P1 = SX:P2 = SY
1500 Y = 12.4: GOSUB 20:Q1 = SX:Q
     2 = SY
1510 X = 0: GOSUB 20:R1 = SX:R2 =
     SY
1520 SL = (Q2 - P2) / (Q1 - P1)
1530  HCOLOR= 4
1540  FOR X = P1 TO Q1
1550 Y = SL * (X - P1) + P2
1560  HPLOT X,T TO X,Y
1570  NEXT X
```

```
1580 SL = (R2 - Q2) / (R1 - Q1)
1590  FOR X = Q1 TO R1:Y = SL * (
     X - Q1) + Q2
1600  HPLOT X,T TO X,Y: NEXT X
1610  REM  MONITOR
1620  HCOLOR= 3
1630 X = 8.75:Y = 3.4:Z = 4: GOSUB
     70
1640 Y = 12.4: GOSUB 100
1650 Z = 10: GOSUB 100
1660 Y = 3.4: GOSUB 100
1670 Z = 4: GOSUB 100
1680  IF T <  > 0 THEN 1780
1690  REM  DIALS
1700 X = 8.75:Y = 11:Z = 6.5: GOSUB
     20
1710  FOR TH = 0 TO 6.3 STEP .4
1720  HPLOT SX,SY TO SX + 4 *  COS
     (TH),SY + 4 *  SIN (TH)
1730  NEXT TH
1740 Z = 8: GOSUB 20
1750  FOR TH = 0 TO 6.3 STEP .4
1760  HPLOT SX,SY TO SX + 4 *  COS
     (TH),SY + 4 *  SIN (TH)
1770  NEXT TH
1780 Y = 3.9:Z = 4.5: GOSUB 20:P1
     = SX:P2 = SY
1790 Y = 9.9: GOSUB 20:Q1 = SX:Q2
     = SY
1800 Z = 9.5: GOSUB 20:R1 = SX:R2
     = SY
1810 Y = 3.9: GOSUB 20:T1 = SX:T2
     = SY
1820  HPLOT P1,P2 TO Q1,Q2 TO R1,
     R2 TO T1,T2 TO P1,P2
1830 Z = 4:Y = 12.4: GOSUB 70
1840 X = 0: GOSUB 100
1850 Z = 10: GOSUB 100
1860 Y = 3.4: GOSUB 100
1870 X = 8.75: GOSUB 100
1880 Y = 12.4: GOSUB 70
1890 X = 0: GOSUB 100
1900  IF T <  > 0 THEN 1940
1910  REM  RETURN FOR SECOND IMAG
     E
1920 T = 40: RESTORE :RHO = 435:C
     X = 140:CY = 64: GOTO 310
1930  REM  FAKE THIRD IMAGE
1940  HPLOT 145,55 TO 149,57 TO 1
     49,59 TO 143,58 TO 143,55
1950  HPLOT 144,56 TO 144,52 TO 1
     48,53 TO 148,57
```

5-4 하노이의 탑(Towers of Hanoi)

고전 수수께끼 「하노이 탑(Towers of Hanoi)」의 게임은 5개의 다른 크기의 원반과 3개의 말뚝을 사용한다. 처음에는 탑 1에 5개의 원반이 크기 순서대로 쌓여 있다. 이 5개의 원반을 탑 3에 이동하는 게임으로 매우 신중하게 진행하여야 한다. 게임을 하다보면 당신은 경우의 수라는 계산방법을 생각하게 될 것이다. 경우의 수와 비슷하게 진행되는 이 게임은 조건이 따라 붙는다. 그 하나가 이동하는 원반은 그 이동되는 원반보다 길이가 짧은 원반위에는 절대로 옮길 수 없다. 또 한번에 원반 한개만 옮길 수 있다.

이제 게임을 시작해 보자. RUN를 타이프하고 RETURN 키를 누르면 그림 5-5가 나타나고, 텍스트 부분에는

```
FROM TOWER,TO TOWER?
```

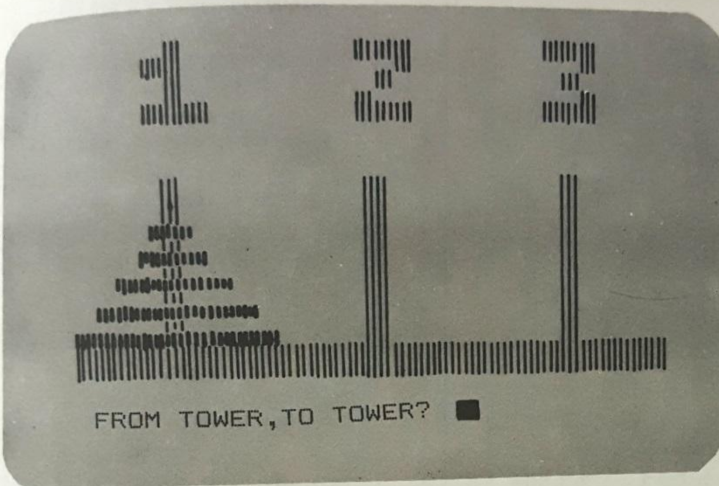


그림 5 - 5

로서'어느탑의 원반을 어느 탑으로'하고 묻게된다. 이때 탑1의 제일 작은 원반을 탑2로 옮기고 싶으면 1, 2라고 타이핑하고 RETURN 키를 누른다. 다시 똑 같은 질문을 컴퓨터가 하게된다. 예로써 1, 3이라고 타이핑하고 RETURN 키를 누르면 두번째로 적은 원반이 탑1에서 탑3으로 옮겨간다.

이런 방법으로 탑1의 원반들을 모두 탑3으로 옮기게 되면

GOOD SHOW OL' CHAP!

라는 메세지를 내며 끝냈음을 축하하게 된다.

```
1  REM  ************************
2  REM  *                      *
3  REM  * (GAME)               *
4  REM  *   TOWERS OF HANOI    *
5  REM  *                      *
6  REM  ************************
7  REM
8  REM
20  REM      GRAPHICS CHAPTER
30  REM
40  REM  PLOT THE BASE AND PEGS I
     N GREEN
50  REM
60  HOME : GR : COLOR= 12: DIM A(
     3,5)
70  FOR X = 33 TO 35: HLIN 2,36 AT
     X: NEXT X
80  VLIN 20,33 AT 8
90  VLIN 20,33 AT 19: VLIN 20,33 AT
     30
91  VLIN 6,11 AT 8
92  HLIN 18,20 AT 6: HLIN 18,20 AT
     11: VLIN 6,8 AT 20
93  HLIN 29,31 AT 6: HLIN 29,31 AT
     11: VLIN 9,11 AT 31
94  PLOT 19,9: PLOT 18,10: PLOT 3
     1,7: PLOT 30,8
95  HLIN 7,9 AT 11: PLOT 7,7
100  REM
110  REM  INITIALIZE 3 ARRAYS FOR
      EACH PEG
120  REM
130  COLOR= 1: FOR X = 1 TO 5
140  HLIN 8 - X,8 + X AT 2 * X +
      22
150 A(1,X) = X:A(2,X) = 0:A(3,X) =
      0
160  NEXT X
```

```
170  REM
180  REM  MOVEMENT OF DISKS
190  REM
200  INPUT "FROM TOWER,TO TOWER?
     ";T1,T2
210 DISK = 0: FOR X = 1 TO 5
220  IF DISK > 0 OR A(T1,X) = 0 THEN
     260
230 DISK = A(T1,X):XCTR = 8 + (T1
     - 1) * 11:A(T1,X) = 0
240  COLOR= 0: HLIN XCTR - DISK,X
     CTR + DISK AT 2 * X + 22
250  COLOR= 12: PLOT XCTR,2 * X +
     22: COLOR= 1
260  NEXT X
270  IF DISK = 0 THEN  PRINT "NO
     DISK THERE!": GOTO 200
280 EMPTY = 0: FOR X = 5 TO 1 STEP
     - 1
```

```
290  IF EMPTY = 0 AND A(T2,X) = 0
      THEN EMPTY = X
300  NEXT X: IF EMPTY = 5 THEN 33
     0
310  IF A(T2,EMPTY + 1) >  = DISK
      THEN 330
320  PRINT "YOU CAN'T DO THAT!":T
     2 = T1: GOTO 280
330  REM  PLOT THE DISK AT A(T2,E
     MPTY)
340 X = 8 + (T2 - 1) * 11:Y = 2 *
     EMPTY + 22
350  COLOR= 1: HLIN X - DISK,X +
     DISK AT Y
360 A(T2,EMPTY) = DISK
370  PRINT : IF T2 = 1 OR EMPTY <
      > 1 THEN 200
380  PRINT "GOOD SHOW OL' CHAP!"
```

5-5 공중선 격추게임(Drizzle)



이 게임은 아군의 고사포로 적의 공중선을 격추시키는 게임이다. 게임방법은 적의 공중선이 왼쪽에서 오른쪽으로 날아갈때 **SPACE** 키를 눌러 포를 발사시켜 격추시킨다. 이때 그외의 다른 키를 눌러 포가 올라가는 것을 멈추게 할 수도 있다. 공중선이 지나가는 댓수에 대해 포의 발사수가 많으면 격추시에 그 만큼 성적이 나빠진다. 즉성적은 백분율로서 나타난다.

적의 공중선이 격추되면 폭발음을 내면서 떨어지고 완전히 지상에 내려오면 버섯구름이 나타나면서 공중선내부의 가스폭발을 알린다.

저해상도 그래픽스로 구성된 이 프로그램에서 화면에 표시되는 컬러를 여러가지로 변화하도록 하므로써 구성도형에 대한 단조로움을 색의 변화로 그만큼 감했다.

```
9 REM ***********************
10 REM *                    *
11 REM * ( GAME )           *
12 REM *  D R I Z Z L E     *
13 REM *                    *
14 REM ***********************
15 REM
20 GOSUB 1000: HOME : INVERSE
40 SWEAR$ = "  @#&!  "
50 Q =  - 16384:BUZZ = Q + 48:BEE
   P$ =  CHR$ (7)
70 GR
80 VTAB 23: HTAB 1: PRINT ":SAUC
   ERS:";: HTAB 11: PRINT "::SH
   OTS::";: HTAB 21: PRINT "::H
   ITS:::";: HTAB 31: PRINT ":P
   ERCENT:"
85 VTAB 24: HTAB 1: PRINT ":::::
   ::::";: HTAB 11: PRINT "::::
   :::::";: HTAB 21: PRINT ":::
   ::::::";: HTAB 31: PRINT "::
   :00%:::";
88 GOSUB 370:SLUGHT = 35:XFALL =
   1:YY = 0:HRZ = 0:SCRCLR =  INT
   ( RND (1) * 14) + 1: GOSUB 8
   00
90 VTAB 21: HTAB 1
95 PRINT " SPACE TO FIRE, ANY KE
   Y TO STOP BULLET "
97 GOSUB 800
100 REM
        G U N
101 REM
110 GUNCLR =  INT ( RND (1) * 14)
    + 1: IF GUNCLR = SCRCLR THEN
    110
115 COLOR= GUNCLR: HLIN 22,26 AT
    39: HLIN 23,25 AT 38: VLIN 3
    6,37 AT 24
120 K =  PEEK (Q): IF K <  > 160 THEN
    220
130 IF SLUGHT <  > 35 THEN 140
132 SHOT = SHOT + 1: GOSUB 610
135 GOSUB 670
140 COLOR= 15
150 IF  SCRN( 24,SLUGHT - 1) = 0
    THEN 200
151 REM
        H I T !
152 REM
153 PLOT 24,SLUGHT - 1: COLOR= 0
    : PLOT 24,SLUGHT
161 COLOR= 9
162 PLOT 22,SLUGHT - 3: PLOT 23,
    SLUGHT - 2: PLOT 23,SLUGHT: PLOT
    24,SLUGHT - 1: PLOT 25,SLUGH
    T: PLOT 27,SLUGHT - 2: VLIN
    SLUGHT - 3,SLUGHT - 2 AT 25
166 FOR I = 1 TO 9: GOSUB 620: NEXT
    :KILL = 1
170 COLOR= 0: FOR I = 22 TO 27: VLIN
    SLUGHT - 3,SLUGHT AT I: NEXT
180 HIT = HIT + 1: GOSUB 670
190 YY = 1:XFALL = 0
200 PLOT 24,SLUGHT - 1: COLOR= 0
    : PLOT 24,SLUGHT
210 SLUGHT = SLUGHT - 1: IF SLUGH
    T > YFALL - 3 THEN 220
211 COLOR= 0: VLIN 0,35 AT 24:SL
    UGHT = 35: POKE Q + 16,0
220 GOSUB 240
230 GOTO 120
240 REM
        SAUCER
241 REM
250 COLOR= SCRCLR
270 HLIN HRZ + 2,HRZ + 3 AT YFAL
    L: HLIN HRZ + 1,HRZ + 4 AT Y
    FALL + 1
280 COLOR= 0
290 PLOT HRZ + 1,YFALL: PLOT HRZ
    ,YFALL + 1
300 IF KILL THEN  PLOT HRZ + 4,Y
    FALL: HLIN HRZ,HRZ + 5 AT YF
    ALL - 1: HLIN HRZ,HRZ + 5 AT
    YFALL - 2
305 IF YY = 0 THEN 310
306 N = 41 - YFALL: GOSUB 900
307 XFALL = XFALL +  INT ( RND (1
    ) * 3) - 1: IF HRZ + XFALL <
    19 THEN XFALL = 1
308 IF HRZ + XFALL > 25 THEN XFA
    LL =  - 1
310 HRZ = HRZ + XFALL:YFALL = YFA
    LL + YY: IF HRZ > 35 THEN 34
    0
320 IF YFALL > 38 THEN  GOSUB 39
    0
```

```
330  RETURN
340  HLIN 36,39 AT YFALL + 1: HLIN
     37,38 AT YFALL
360  GOSUB 800:SCRCLR =  INT ( RND
     (1) * 14) + 1: IF SCRCLR = G
     UNCLR THEN 360
370 SAUCER = SAUCER + 1: GOSUB 67
     0
380 HRZ = 0: RETURN
390  REM
        EXPLOSION
391  REM
400  VTAB 21: HTAB HRZ: PRINT SWE
     AR$:KILL = 0
420  COLOR= 15
422  PLOT HRZ,37: PLOT HRZ + 1,38
     : VLIN 35,37 AT HRZ + 2: PLOT
     HRZ + 2,39: VLIN 38,39 AT HR
     Z + 3: VLIN 36,37 AT HRZ + 4
     : PLOT HRZ + 5,37
425  POP : PRINT BEEP$;BEEP$;BEEP
     $;BEEP$;
427  COLOR= SCRCLR
430  FOR I = 39 TO 6 STEP  - 2
450  HLIN 19,29 AT I - 1
460  HLIN 19,29 AT I: GOSUB 640
470  NEXT
480  HLIN 29,33 AT 2: HLIN 20,23 AT
     3: HLIN 28,34 AT 3: HLIN 18,
     24 AT 4: HLIN 27,35 AT 4: HLIN
     17,35 AT 5: HLIN 16,35 AT 6:
      HLIN 15,35 AT 7
490  HLIN 16,36 AT 8: HLIN 13,37 AT
     9: HLIN 12,38 AT 10: HLIN 11
     ,38 AT 11: HLIN 11,38 AT 12:
      HLIN 10,38 AT 13: HLIN 10,3
     8 AT 14: HLIN 10,37 AT 15
500  HLIN 10,37 AT 16: HLIN 10,37
      AT 17: HLIN 11,14 AT 18: HLIN
     16,36 AT 18: HLIN 12,13 AT 1
     9: HLIN 16,34 AT 19: HLIN 16
     ,34 AT 20: HLIN 17,34 AT 21
510  HLIN 31,33 AT 22
540  FOR I = 1 TO 22
545  IF I -  INT (I / 2) * 2 = 0 THEN
      COLOR= SCRCLR
546  IF I -  INT (I / 2) * 2 = 1 THEN
      COLOR= SCRCLR
```

```
550 HRZ =  INT ( RND (1) * 40):YF
     ALL =  INT ( RND (1) * 20) +
     20: GOSUB 640: PLOT HRZ,YFAL
     L: NEXT
566  COLOR= 0
570  FOR I = 0 TO 39: HLIN 0,39 AT
     I: FOR J = 1 TO 9: NEXT : NEXT
     : FOR J = 1 TO 222: NEXT
580  POP : GOTO 88
590  FOR I = 1 TO 500: NEXT
600  REM
        NOISE
601  REM
610  COLOR= 9: HLIN 23,25 AT 35: HLIN
     22,23 AT 34: HLIN 25,26 AT 3
     4
620  FOR J = 1 TO 10:S =  PEEK (B
     UZZ) -  PEEK (BUZZ) -  PEEK
     (BUZZ): NEXT
630  COLOR= 0: HLIN 23,25 AT 35: HLIN
     22,26 AT 34: RETURN
640  FOR J = 1 TO 2
650 S =  PEEK (BUZZ) -  PEEK (BUZ
     Z) -  PEEK (BUZZ) -  PEEK (B
     UZZ) -  PEEK (BUZZ) -  PEEK
     (BUZZ) -  PEEK (BUZZ): NEXT
     : RETURN
660  REM
        SCOREBOARD
661  REM
670  VTAB 24: HTAB 4: PRINT SAUCE
     R;: HTAB 14: PRINT SHOT;
680  HTAB 24: PRINT HIT;: IF  NOT
     HIT THEN  RETURN
690  HTAB 34: PRINT  INT ((100 *
     HIT) / SAUCER);"%:";
700  RETURN
800 YFALL = Y1:Y1 =  INT ( RND (1
     ) * 25) + 3: RETURN
900  POKE 0,100 - (3 * N): POKE 1
     ,5
902  CALL 771
904  RETURN
1000  FOR I = 771 TO 789: READ A:
      POKE I,A: NEXT : RETURN
1100  DATA  173,48,192,136,208,4,
      198,1,240,8,202,208,246,166,
      0,76,3,3,96
```

5-6 미로(Maze)

저해상도 그래픽을 이용한 미로 발생의 조직을 연구해 보자.

프로그램을 입력시키고 RUN을 타이프하고 RETURN 키를 누르면 하단에는「GENERATING MAZE」 즉 미로 발생중이라는 표시가 나오고 그래픽 부분에는 블록들이 계속 생기거나 없어지기도 한다. 한번 모두 없어지고 다시 같은 일이 반복된다. 이때까지 약 30초가 걸린다.

화면이 모두 지워지고 중앙에 미로의 출발점과 사방 한 단계씩의 벽과 길이 나타난다. 당신은 미로에 들어섰다. 빠져나올 문은 좌측 제일 상단에 있다. 암흑속을 전진하면서 당신이 볼 수 있는 것은 전후좌우 일부분이다.

이제 당신은 미로를 헤쳐 나간다. I 는 윗쪽, J 는 왼쪽, K 는 오른쪽, M 은 아랫쪽을 지시한다. 이에 따라 계속 상하좌우로 한 걸음 밖의 시야가 나타난다. 도중에는 갈림길이 나타나고 당신은 빠르게 결단을 내려야 한다.

목표는 빨리 빠져나올 문에 도달하는 것이다. 목적지에 도달하면「YOU DID IT」이라는 메세지가 껌뻑이며 나타나고 소요시간이 점수로 나타난다. 당연히 점수는 적을수록 잘한 것이다. 전진하다가 미궁에 빠지면 스페이스바를 눌러 전체 미로의 모습을 잠깐 볼 수 있다. 하지만 한번 스페이스바를 누를 때마다 5백점이 가산되고, 여태까지 밟아왔던 길의 모습까지 모조리 시야에서 사라져, 남는 것은 다시 당신 주변의 상하좌우 한걸음 뿐이다. 따라서 스페이스 바를 눌러 전체 모습을 보는 것은 되도록 삼가해야 된다.

문에 도달하여 당신이 미로를 빠져 나오는 것을 해냈다는 메세지를 보는 순간 당신은 큰 해방감을 맛볼 것이다.

이 프로그램을 연구하면 휠씬 세련된 고해상도그래픽 화면에 다양한 아이디어를 첨가할 수 있을 것이다.



그림 5-6 미 로

```
1 REM ************************
2 REM *                      *
3 REM *   GAME : M A Z E     *
4 REM *                      *
5 REM ************************
6 REM
110 HOME
120 GR
130 HTAB 13: FLASH : PRINT "GENE
     RATING MAZE": NORMAL
150 A(0) = 2:A(1) = - 80:A(2) =
     - 2:A(3) = 80
160 WL = 8:HL = 7:CT = 0
170 DIM M(1680)
180 A = 859:L = 40
200 FOR I = 2 TO 38: FOR J = 0 TO
     38
210 M(I * L + J) = WL: COLOR= J: PLOT
     L * RND (1),L * RND (1): NEXT
     J,I
220 M(A) = 4: GR
230 J = INT ( RND (1) * 4):Z = J

240 COLOR= 16 * RND (1)
250 PLOT L * RND (1),L * RND (
     1)
260 TT = PEEK ( - 16336)
270 B = A + A(J): IF M(B) = WL THEN
     M(B) = J:M(A + A(J) / 2) = H
     L:A = B: GOTO 230
280 J = (J + 1) * (J < 3): IF J <
     > Z THEN 270
290 J = M(A):M(A) = HL: IF J < 4 THEN
     A = A - A(J): GOTO 230
300 GR :PX = 19:PY = 21: HOME
310 XD(0) = 1:YD(0) = 0:XD(1) = 0
     :YD(1) = 1:XD(2) = - 1:YD(2
     ) = 1:XD(3) = 1:YD(3) = 1
320 D(0) = 1:D(1) = 40:D(2) = 39:
     D(3) = 41
340 FOR I = 1 TO 20:M((36 * RND
     (1) + 2) * 40 + 38 * RND (1
     )) = HL: NEXT
350 LOC = PX + PY * 40
360 FOR I = 0 TO 3
370 COLOR= M(LOC + D(I)): PLOT P
     X + XD(I),PY + YD(I)
380 COLOR= M(LOC - D(I)): PLOT P
     X - XD(I),PY - YD(I)
390 NEXT
400 COLOR= 15: PLOT PX,PY
410 DIR = PEEK ( - 16384): IF DI
     R < 128 THEN CT = CT + 1: GOTO
     410
420 POKE - 16368,0:T = PEEK ( -
     16336)
430 DIR = DIR - 128
440 TX = PX + (DIR = 75) - (DIR =
     74)
450 TY = PY + (DIR = 77) - (DIR =
     73)
460 IF DIR = 32 THEN GOSUB 620:
     GOTO 350
470 IF M(TX + TY * 40) < > HL THEN
     410
480 M(TX + TY * 40) = 15:M(PX + P
     Y * 40) = HL
490 COLOR= HL: PLOT PX,PY: COLOR=
     15: PLOT TX,TY:PX = TX:PY =
     TY
500 IF (PX + PY * L) < > 121 THEN
     350
510 FOR K = 1 TO 10
520 FOR I = 1 TO 10:A = PEEK ( -
     16336): FOR W = 1 TO 10 - I:
     NEXT W,I
530 FOR W = 1 TO 50: NEXT : NEXT

540 TEXT : HOME : FLASH
550 FOR I = 1 TO 24: PRINT "


                                  "
560 NEXT : VTAB 11: HTAB 15: PRINT
     " YOU DID IT !"
570 FOR W = 1 TO 5000: NEXT
580 HOME : INVERSE
590 PRINT "YOUR SCORE : ";CT
600 NORMAL
610 END
620 :
630 FOR I = 2 TO 39
640 FOR J = 0 TO 38
650 COLOR= M(I * L + J): PLOT J,
     I
660 NEXT J
670 NEXT I
680 FOR W = 1 TO 500: NEXT
690 CT = CT + 500: GR : RETURN
```

5-7 포쏘기 (Super artillery)

이 게임은 아군의 포로 적의 포를 파괴하여 점수를 얻는 게임이다. 게임의 특징은 두뇌를 재치있게 움직이지 않으면 상대방을 이기기가 거의 불가능하다는 것이다.

이 게임을 RUN시켜 수행하면 시작음을 내면서 화면에는 다음과 같은 설명이 나타난다.

```
]
        *** SUPER ARTILLERY ***

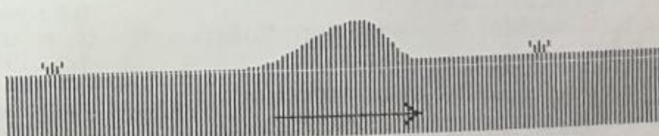
GAME RULES ARE SIMPLE:
BLOW UP YOUR OPPONENT BY SHOOTING AT THE
PROPER ANGLE AND VELOCITY,

WHILE COMPENSATING FOR WIND AND TERRAIN.

EACH GUNNER MUST ENTER THE GUN ELEVATION
AND THE SHELL VELOCITY.

THE ELEVATION MUST BE BETWEEN 0 FOR A
HORIZONTAL SHOT, AND 90 FOR STRAIGHT UP.
PLAYER 1, WHAT IS YOUR NAME? A
PLAYER 2, WHAT IS YOUR NAME? B
HOW MANY TOTAL GAMES DO YOU WISH TO PLAY10
```

게임할 사람의 이름-PLAYER 1, PLAYER 2 -을 입력하면 몇 게임을 할 것인가를 컴퓨터가 묻게 되는데 예를 들어 10을 입력하면 게임은 앞으로 10전이 된다.



```
THE BATTLE IS ABOUT TO COMMENCE . . .
        WIND AT 39
```

그림 5-7

다음은 방어막이 나타나면서 바람의 방향과 풍속을 표시하는데 이것은 순간적으로 사라진다. 그러면서 지금까지의 게임성적이 나타난다.

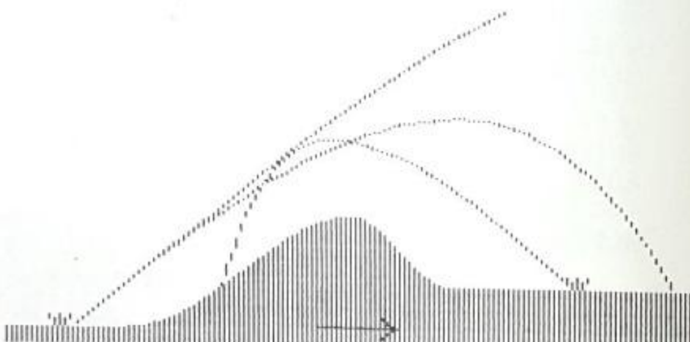


```
ONE CASTLE FOR EACH OF YOU....

THE SCORE:  A        =0
            B        =0
```

그림 5 - 8

다음은 게임이 시작된다. PLAYER 1이 먼저 공격을 하게 된다. 각도와 속도를 입력하므로 포에서 포탄이 공중으로 날아간다. 이 흔적은 게임이 끝날 때까지 그대로 있으므로 이것을 기준으로 하여 앞으로 게임에서 각도와 속도를 계산하는데 참고할 수 있을 것이다.



```
A          45  45 0   0              B
LAST SHOTS 100 150 0   0
CALL YOUR SHOT - (ELEVATION, VELOCITY)? 45,130
```

그림 5 - 9

이런식으로 거리조정과 각도가 어느정도 상대방의 포에 접근되다가 적중되면 상대의 포는 폭발하면서 1게임이 끝난다. 상대의 포를 폭발시킬 때의 기분을 돋구기 위해 스피커에서는 신나는 폭발음이 일어난다. 자, 1전은 끝났다. 다시 점수가 나타나고 게임의 2전이 시작된다. 방어막은 난수에 의해 결정되므로 산의 모양은 아까와는 딴판이다. 그러므로 새로운 계산에 의해 게임을 하지 않으면 안된다. 또 하나 주의할 것은 PLAYER 1의 계산과 PLAYER 2의 계산방식은 다르다는 것이다. 왜냐하면 언제든지 한쪽으로 거센 바람이나 혹은 미풍이 불고 있기 때문에 그 바람의 속도에 따라 포의 진행속도가 달라지기 때문이다.

249 쪽 파본

250 쪽 파본

```
    *** SUPER ARTILLERY ***": HTAB
    21 - LEN (A$) / 2
730 FOR I = 1 TO LEN (A$): PRINT
     MID$ (A$,I,1);:Z = PEEK ( -
    16336) + PEEK ( - 16336) +
     PEEK ( - 16336) + PEEK ( -
    16336): NEXT
740 FOR I = 1 TO 2000: NEXT I
750 HOME : VTAB 3:A$ = "GAME RUL
    ES ARE SIMPLE:": GOSUB 990: PRINT

760 A$ = "BLOW UP YOUR OPPONENT B
    Y SHOOTING AT THEPROPER ANGL
    E AND VELOCITY,": GOSUB 990
770 PRINT :A$ = "WHILE COMPENSAT
    ING FOR WIND AND TERRAIN.": GOSUB
    990
780 A$ = "EACH GUNNER MUST ENTER
    THE GUN ELEVATIONAND THE SHE
    LL VELOCITY.": GOSUB 990: PRINT

790 A$ = "THE ELEVATION MUST BE B
    ETWEEN 0 FOR A   HORIZONTAL
    SHOT, AND 90 FOR STRAIGHT UP
    .": GOSUB 990
800 VTAB 21: POKE 34,20
810 FOR P = 1 TO 2
820 PRINT "PLAYER ";P;: INPUT ",
     WHAT IS YOUR NAME? ";P$(P):
     IF LEN (P$(P)) > 10 THEN HOME
    : PRINT "PLEASE USE A NAME L
    ESS THAN 10 LETTERS": GOTO 8
    20
830 HOME : NEXT
840 DEF FN D(X) = INT (X * RND
    (1))
850 INPUT "HOW MANY TOTAL GAMES
    DO YOU WISH TO PLAY";GN
860 IF GN < 1 THEN PRINT "PLEAS
    E, DON'T JOKE AROUND!": GOTO
    850
870 POKE 232,72: POKE 233,3: SCALE=
    1: ROT= 0
880 FOR I = 0 TO 19
890 READ J
900 POKE I + 768,J: NEXT
910 DATA 160,0,136,208,253,173,
    48,192,206,1,3,206,1,3,206,1
    ,3,208,237,96
920 FOR I = 0 TO 39
930 READ J
940 POKE I + 840,J
950 NEXT : FOR I = 1 TO 1500: NEXT
    I: HOME : GOTO 70
960 DATA 1,0,4,0
970 DATA 109,56,63,63,44,45,45,
    13,8,63,31,63,63,56,56,56
980 DATA 48,146,45,69,41,33,33,
    33,33,219,23,23,18,36,35,12,
    56,155,4,0
990 FOR I = 1 TO LEN (A$): PRINT
     MID$ (A$,I,1);
1000 Z = PEEK ( - 16336): NEXT :
     PRINT
1010 FOR I = 1 TO 700: NEXT I
1020 RETURN
1030 IF PEEK (222) < > 255 THEN
     RESUME
1040 TEXT : HOME : VTAB 10
1050 PRINT "THE SCORE:  ";P$(1);
    : HTAB 23: PRINT " =";S(1): HTAB
    13: PRINT P$(2);: HTAB 23: PRINT
    " =";S(2)
1060 IF S(1) > S(2) THEN WI = 1
1070 IF S(1) < S(2) THEN WI = 2
1080 IF S(1) = S(2) THEN 1120
1090 VTAB 9 + WI: HTAB 28
1100 INVERSE : PRINT "WINNER"
1110 GOTO 1140
1120 PRINT : HTAB 13: INVERSE
1130 PRINT " TIE "
1140 NORMAL : VTAB 20
1160 PRINT "BETTER LUCK AT THE
                          PEACE TA
    LKS NEXT TIME!!"
1170 INPUT "DO YOU WANT TO PLAY
    AGAIN (Y/N)? ";A$
1180 IF A$ = "Y" THEN RESTORE :
     HOME : POKE 34,20: HOME : FOR
    I = 1 TO 10:A(I) = 0:V(I) =
    0: NEXT :S(1) = 0:S(2) = 0: GOTO
    850
1190 IF A$ < > "N" THEN PRINT
     CHR$ (7);: VTAB PEEK (37):
     GOTO 1170
1200 TEXT : HOME : NEW : END
```

5-8 야구게임(Base Ball)

패들이나 조이스틱을 이용하여 야구게임을 해 보자. 모든 규칙은 야구경기와 같다. 수비와공격을 정해 서로 상대방끼리 누가 더 점수를 올리는가 시합을 해 보아도 좋을 것이다.

이 게임은 2사람이 같이 할 수 있는 게임인데 팀을 짜서 여러명이 할 수도 있다. 게임규칙은 야구와 거의 동일하다. 단, 수비가 피쳐 한 사람만 화면에 나타나는 것이 다른 점이다.

게임을 RUN시키면 화면에는 그림 5-11과 같이 야구장이 펼쳐진다. 자, 죠이스틱을 움직여 게임을 하자.

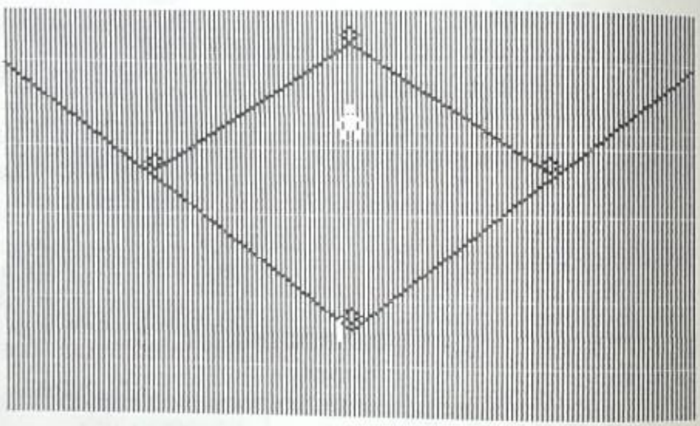


그림 5-11

PDL(0)와 PDL(1)로 서로 상대방에 대해 공격수비를 하자. 첫번째 죠이스틱의 PDL(1)을 누르면 피쳐는 공을 던지게 된다. 이때 공격자는 날아오는공이 스트라이크존을 통과할때를 맞추어 PDL(0)를 누르면 된다.

히트시 수비가 나타나 공을 받을 수 있으며 이것은 아웃이 된다.

공이 제대로 스트라이크존안에 들어오지 않을 경우 볼로 나타나고 히트되었을 경우에는 장외 아웃, SINGLE(1루타), DOUBLE(2루타), TRIBLE(3루타), HOMERUN(홈런)이 되며 그림 5-12와 같이 해당히트에 맞게 주자가 뛰게 된다.

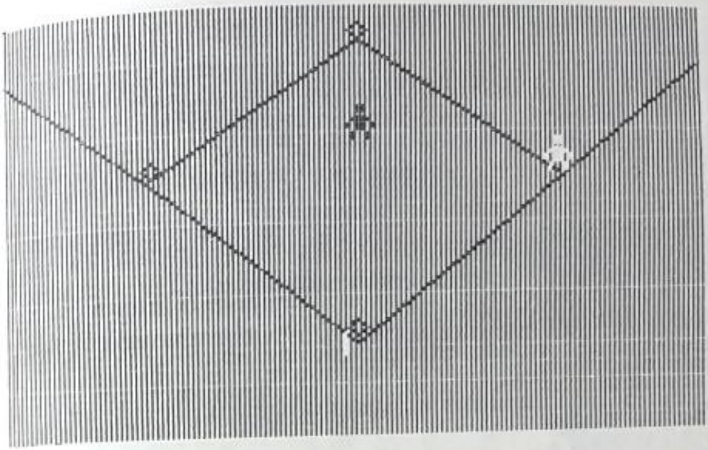


그림 5-12

```
0  REM ************************
1  REM *                      *
2  REM * (GAME)  BASE  BALL   *
3  REM *                      *
4  REM ************************
5  GOTO 6100
10  LOMEM: 17434: HOME : GOSUB 23
     20: POKE 232,0: POKE 233,64:
      DIM S(18)
15 IN = 1
20  IF B = 2 THEN B = 1:R = 2: GOTO
     370
40 B = 2:R = 1: GOTO 370
50  IF IN / 2 =  INT (IN / 2) THEN
     RE = RE + E: GOTO 70
60 BE = BE + E
70 E = 0:RO = 0:X4 = 0: IF HT = 1
     THEN  GOSUB 790: GOTO 50
80  IF BL = 4 THEN  GOSUB 120: GOSUB
     1690
90  IF ST = 3 THEN  GOSUB 1980: PRINT
     "YER OUT!!":OT = OT + 1: GOSUB
     120: GOSUB 2270
100  IF OT > 2 THEN OT = 0:B1 = 0
     :B2 = 0:B3 = 0:S(IN) = P:P =
     0: GOSUB 120: GOTO 130
110  GOTO 370
120 BL = 0:ST = 0:HT = 0: RETURN

130  POKE 34,0: TEXT : HOME : PRINT
      SPC( 10): FLASH : PRINT "AL
     STAR BASEBALL": NORMAL
140  PRINT : PRINT : PRINT : PRINT
     "INNING": PRINT : PRINT : PRINT
     "VISTORS": PRINT : PRINT "HO
     ME"
150  FOR I = 1 TO 9: HTAB I * 3 +
     9: VTAB 5: PRINT I;: NEXT I
160 BS = 0:RS = 0: FOR I = 1 TO I
     N
170  IF I / 2 =  INT (I / 2) THEN
     210
180 RS = RS + S(I): HTAB (I / 2 *
     3 + 11): VTAB 8
190  PRINT S(I);: NEXT I
```

```
200  GOTO 220
210 BS = BS + S(I): HTAB I / 2 *
     3 + 9: VTAB 10: PRINT S(I);:
      NEXT I
220  HTAB 1: VTAB 14: PRINT "BOXS
     CORE:RUNS HITS ERRORS HIT%";

230  HTAB 2: VTAB 16: PRINT "VIST
     ORS": HTAB 1: VTAB 18: PRINT
     "HOME"
240  HTAB 13: VTAB 16: PRINT RS: HTAB
     19: VTAB 16: PRINT RH: HTAB
     26: VTAB 16: PRINT RE
250  HTAB 13: VTAB 18: PRINT BS: HTAB
     19: VTAB 18: PRINT BH: HTAB
     26: VTAB 18: PRINT BE
255  IF RA = 0 THEN  GOTO 290
260 PE =  INT (RH / RA * 1000): HTAB
     31: VTAB 16: PRINT ".";PE
270  IF BA = 0 THEN 290
280 PE =  INT (BH / RA * 1000): HTAB
     31: VTAB 18: PRINT ".";PE
290  HTAB 9: VTAB 23: PRINT "PRES
     S BUTTON TO CONTINUE...";
295  IF IN = 18 THEN 310
300  IF  PEEK ( - 16287) < 128 AND
      PEEK ( - 16286) < 128 THEN
     300
310 IN = IN + 1: IF IN = 19 THEN
     340
320  IF IN = 18 AND BS > RS THEN
     340
330  GOTO 20
340  FOR I = 1 TO 100: NEXT I: HTAB
     9: VTAB 23: PRINT "PRESS BUT
     TON FOR NEXT GAME?";
350  IF  PEEK ( - 16287) > 127 OR
      PEEK ( - 16287) > 127 THEN
      CLEAR : GOTO 20
360  GOTO 350
370  HGR : HCOLOR= 1: HPLOT 0,0: CALL
     62454
380 PT = R + 1: IF R = 1 THEN FT =
     0: GOTO 400
390 FT = 3
```

```
400  HCOLOR= 3: HPLOT 0,25 TO 140
     ,150 TO 279,25
405  HPLOT 60,75 TO 140,15 TO 220
     ,75: GOSUB 1750
410  DRAW 2 AT 140,147
420 P3 = 133: IF FT = 3 THEN P3 =
     134
430  HCOLOR= FT: ROT= 0: SCALE= 1
     : DRAW 3 AT P3,44
440  GOSUB 1980
450  PRINT "RUNS  ";P;"   OUTS  "
     ;OT
460  PRINT "BALLS  ";BL;"   STRIK
     ES";ST;: PRINT
470  SCALE= 11: HCOLOR= 0:BR = 64
     : ROT= BR: XDRAW 1 AT 135,14
     5
480 A$ = "VISTORS": IF B = 2 THEN
     A$ = "HOME"
490  PRINT A$;"S PITCH   ";
500  IF  PEEK (PT - 16289) < 128 THEN
     500
510 XY = 141:YX = 60
520  IF  PDL (PT - 2) >  - 1 AND
     PDL (PT - 2) < 95 THEN SP =
     2.5: GOTO 550
530  IF  PDL (PT - 2) > 159 AND  PDL
     (PT - 2) < 256 THEN SP = 7: GOTO
     550
540 SP = 4
550  IF XY < 130 THEN 610
560 CV =  SGN ( INT (( PDL (PT -
     2) - 32) / 191))
590  IF XY = 135 AND CV =  - 1 THEN
     CV = 0
600  IF XY = 148 AND CV = 1 THEN
     CV = 0
610 YX = YX + 2 + SP:XY = XY + CV
620  ROT= 0: SCALE= 1: HCOLOR= 3:
     DRAW 2 AT XY,YX
630  HCOLOR= 1: DRAW 2 AT XY - CV
     ,YX - 2 - SP
640  IF  PEEK (B - 16288) > 127 THEN
     X4 = 1
650  IF X4 = 1 THEN 680
660  IF YX > 138 THEN 700
670  GOTO 550
680 BR = BR - 5: ROT= BR: SCALE=
     11: XDRAW 1 AT 135,145
690  ROT= BR + 5: XDRAW 1 AT 135,
     145
700  IF YX > 138 OR BR < 32 THEN
     720
710  GOTO 550
720  IF YX > 150 AND XY > 137 AND
     XY < 147 THEN ST = ST + 1: GOTO
     50
730  IF BR = 64 AND XY > 146 OR B
     R = 64 AND XY < 137 THEN BL =
     BL + 1: GOTO 50
740  IF BR < 32 THEN ST = ST + 1:
     GOTO 50
750  IF BR <  > 64 AND XY > 146 OR
     BR <  > 64 AND XY < 137 THEN
     ST = ST + 1: GOTO 50
760  IF YX > 137 AND YX < 151 AND
     BR < 57 AND BR > 39 THEN HT =
     1: GOSUB 3000: GOTO 50
770  IF XY > 136 AND YX < 147 THEN
     ST = ST + 1: GOTO 50
780  GOTO 550
```

```
790 Z =  INT ( RND (1) * 9) - 4
800  SCALE= 1: ROT= 0: HCOLOR= 1
810  DRAW 2 AT XY - CV,YX - 2 - S
     P
820  FOR YX = YX TO 15 STEP  - 3
830 XY = XY + Z: IF XY < 3 OR XY >
     275 THEN 860
840  HCOLOR= 3: DRAW 2 AT XY,YX: HCOLOR= 1
850  DRAW 2 AT XY - Z,YX + 3
860  NEXT YX
880  IF  ABS (Z) <  > 4 THEN 900
890  GOSUB 1980: PRINT "FOUL BALL
     !":ST = ST + 1: IF ST = 3 THEN
     ST = 2
895 HT = 0: RETURN
900  GOSUB 120: GOSUB 1980: GOSUB
     2270
910  HCOLOR= 1: HPLOT 0,0: CALL 6
     2454
920  GOSUB 1980:I =  INT ( RND (1
     ) * 5) + 1: ON I GOTO 930,94
     0,950,960,970
930  PRINT "INFIELD FLY!": GOTO 1
     040
940  PRINT "GROUND BALL!": GOTO 1
     040
950  PRINT "LINE DRIVE!": GOTO 98
     0
960  PRINT "POP UP TO OUTFIELD!":
     GOTO 980
970  PRINT "FLY BALL TO OUTFIELD!
     "
980  HCOLOR= 2: HPLOT 0,0: CALL 6
     2454
990  HCOLOR= 1: FOR I = 140 TO 15
     9: HPLOT 0,I TO 275,I: NEXT
     I
1000  HCOLOR= 3: FOR I = 257 TO 2
     60: HPLOT I,103 TO I,139: NEXT
     I
1010 XY =  INT ( RND (1) * 100) +
     25:Z =  INT ( RND (1) * 25) +
     5:P0 = ( INT ( RND (1) * 40)
     + 27) * 2 + R + 1:P2 = 124:
     P1 = 2:YY =  INT ( RND (1) *
     55) + 70
1030  GOTO 1060
1040 P0 = ( INT ( RND (1) * 64) +
     32) * 2 + R:P1 =  INT ( RND
     (1) * 230) + 15:YY = 145:P2 =
     40
1060 Z1 = 4 *  SGN ( INT (( PDL (
     PT - 2) - 10) / 200))
1090  IF P2 <  > 124 THEN YY = YY
     - 5: GOTO 1130
1100  IF P1 < XY THEN Y1 =  - 1: GOTO
     1115
1110  IF P1 > XY + Z THEN Y1 = 2:
     GOTO 1115
1112 Y1 = 0
1115 YY = YY + Y1
1120 P1 = P1 + 5
1130  IF P0 < 20 AND Z1 =  - 4 OR
     P0 > 240 AND Z1 = 4 THEN 118
     0
1140  IF R0 = 0 THEN P0 = P0 + Z1
1180  IF  PEEK (PT - 16289) < 128
     OR R0 = 1 THEN 1390
1190  IF P0 > 235 AND P2 = 124 AND
     Z1 = 4 THEN 1390
1200 R0 = 1
```

```
1210  HCOLOR= 1: IF P2 = 124 THEN
      HCOLOR= 2
1220  DRAW 3 AT P0 - Z1,P2: ROT=
     48:Z1 = Z1 * 8
1260  HCOLOR= FT
1270  IF P2 = 124 AND Z1 = 0 THEN
     1290
1280  GOTO 1330
1290 P3 = 100: ROT= 0
1300  DRAW 3 AT P0,P3
1310  IF YY < 110 OR YY > 120 THEN
     1550
1320  GOTO 1450
1330 P3 = 60: IF P2 = 40 AND Z1 >
     0 THEN P3 = 45: GOTO 1360
1335  IF P2 = 40 THEN 1360
1340 P3 = 143: IF Z1 > 0 THEN P3 =
     129
1360  IF Z1 > 0 THEN  ROT= 16
1370 P0 = P0 + Z1: DRAW 3 AT P0,P
     3
1380  IF YY > 100 AND YY < 110 THEN
     1450
1390  IF YY < 20 OR YY > 134 AND
     P2 <  > 40 THEN  GOSUB 1980:
     GOTO 1400
1395  GOTO 1410
1400  IF  RND (1) < .1 AND RO = 1
     THEN  PRINT "ERROR":E = E +
     1: GOTO 1700
1405  PRINT "BASE HIT": GOSUB 509
     5: GOSUB 2250: GOTO 1700
1410  IF P2 = 124 THEN 1440
1420  IF YY < 56 AND YY > 40 THEN
     1450
1430  GOTO 1550
1440  IF YY > 134 OR YY < 123 THEN
     1550
1450  IF  ABS (P1 - P0 - 5) < 4 THEN
     1480
1470  GOTO 1550
1480  GOSUB 1980: PRINT "OUT!":OT
      = OT + 1: IF P2 = 124 AND B
     3 = 1 AND OT < 2 THEN B3 = 0
     :P = P + 1: GOSUB 1980: PRINT
     "SACRIFICE!!": PRINT "RUNSCO
     RES!"
1490  RETURN
1550  IF P2 = 124 AND P1 > 255 THEN
      GOSUB 2250: GOTO 1990
1560  HCOLOR= FT: IF P2 = 124 AND
     FT = 2 THEN  HCOLOR= 0
1570  SCALE= 1: IF RO = 1 THEN 15
     90
1580  DRAW 3 AT P0,P2
1590  HCOLOR= 3: DRAW 2 AT P1,YY
1600  HCOLOR= 1: IF P2 = 124 THEN
      HCOLOR= 2
1610  IF P2 = 40 THEN 1640
1620  DRAW 2 AT P1 - 5,YY - Y1
1630  GOTO 1660
1640  DRAW 2 AT P1,YY + 5: IF Z1 =
     0 OR RO = 1 THEN 1060
1650  HCOLOR= 1: IF P2 = 124 THEN
      HCOLOR= 2
1660  IF RO = 1 THEN 1090
1670  IF Z1 = 0 THEN 1060
1680  DRAW 3 AT P0 - Z1,P2: GOTO
     1060
1690  GOSUB 1980: PRINT "BATTER W
     ALKED!": GOSUB 1980: GOTO 17
     10
```

```
1700  GOSUB 1980: IF B3 = 1 THEN
     P = P + 1:T = 1:B3 = 0: PRINT
     "RUNSCORES"
1704  IF B2 = 1 THEN B3 = 1:B2 =
     0: PRINT "RUNNER ADVANCES"
1706  IF B1 = 1 THEN B2 = 1
1708 B1 = 1: RETURN
1710  IF B3 = 1 AND B2 = 1 AND B1
      = 1 THEN T = 1:P = P + 1: GOSUB
     2290
1720  IF B2 = 1 AND B1 = 1 THEN B
     3 = 1: PRINT "RUNNER ADVENCE
     ": RETURN
1730  IF B1 = 1 THEN B2 = 1: PRINT
     "RUNNER ADVANCE": RETURN
1740 B1 = 1: RETURN
1750  HCOLOR= 3 - FT: ROT= 0: SCALE=
     1
1770  IF B1 = 1 THEN 1840
1790  IF B2 = 1 THEN 1860
1810  IF B3 = 1 THEN 1880
1820  GOTO 1890
1840  DRAW 3 AT 211 + R,59: GOTO
     1790
1860  DRAW 3 AT 133 + R,13: GOTO
     1810
1880  DRAW 3 AT 55 + R,60
1890  HCOLOR= 3: HPLOT 137,140 TO
     141,140
1900  SCALE= 5: ROT= 8
1910  IF B1 = 1 THEN 1930
1920  DRAW 2 AT 220,75
1930  IF B2 = 1 THEN 1950
1940  DRAW 2 AT 140,15
1950  IF B3 = 1 THEN 1970
1960  DRAW 2 AT 60,75
1970  RETURN
1980  FOR I = 1 TO 100: NEXT I: POKE
     34,20: HOME : RETURN
1990  IF YY > 115 THEN 2100
2000  IF YY < 104 THEN 2200
2010 T = B1 + B2 + B3:P = P + T: PRINT
     "TRIPLE!"
2015  GOSUB 5080
2020 B1 = 0:B2 = 0:B3 = 1: RETURN
2100 T = B2 + B3:P = P + T: PRINT
     "DOUBLE!"
2105  GOSUB 5070
2110 B3 = 0: IF B1 = 1 THEN B3 =
     1
2120 B2 = 1:B1 = 0: RETURN
2200 T = B1 + B2 + B3 + 1:P = P +
     T
2210  IF T = 4 THEN : PRINT "GRAN
     D SLAM!!!!": GOTO 2230
2220  PRINT "HOME RUN!"
2225  GOSUB 5090
2230  PRINT T;"RUNS SCORE!"
2240 B1 = 0:B2 = 0:B3 = 0:T = 0: RETURN
2250  IF IN / 2 =  INT (IN / 2) THEN
     BH = BH + 1: RETURN
2260 RH = RH + 1: RETURN
2270  IF IN / 2 =  INT (IN / 2) THEN
     BA = BA + 1: RETURN
2280 RA = RA + 1: RETURN
2290  GOSUB 1980: PRINT T;"RUN";:
      IF T > 1 THEN  PRINT "S";
2300  PRINT "SCORE";: IF T = 1 THEN
      PRINT "S";
2310  PRINT :T = 0: RETURN
```

```
2320 L = 16384
2330  FOR MX = 1 TO 2
2340  READ A$
2350  FOR I = 1 TO  LEN (A$) STEP
     2
2360 AD =  ASC ( MID$ (A$,I,1)) -
     48
2370  IF AD > 9 THEN AD = AD - 7
2380 CH =  ASC ( MID$ (A$,I + 1,1
     )) - 48
2390  IF CH > 9 THEN CH = CH - 7
2400  POKE L,AD * 16 + CH
2410 L = L + 1: NEXT I
2420  NEXT MX
2430  RETURN
2440  DATA "03000B000A000D0006003
     C2E000909092D0D0909111B1B1B3
     B3F1F1B130909092D0D0909111B1
     B1B3B3F1B1B13090909290909091
     11B1B1B3F3F1F1B1309092D2D"
2450  DATA  "2D0D09111B1B3F3B3F3B
     1F13092D092D0D290D111B3B1B3B
     3F1B3B13090D092D0D090D111B1B
     1B3F3B1F1B1309090D09290D0911
     1B1B3B1B1B3B1B1309090D09090D
     09111B1B3F1B1B3B1F1300"
3000 S =  - 16336
3010  FOR BEEP = 1 TO 5
3020 SOUND =  PEEK (S) -  PEEK (S
     ) +  PEEK (S) -  PEEK (S) +
      PEEK (S) -  PEEK (S)
3030  NEXT BEEP
3040  RETURN
5000  REM  HERE'S SOUND FOR THE G
     AME
```

```
5070 M = 2:N = 2:DE = 7: GOSUB 60
     00: RETURN
5080 M = 30:N = 24:DE = 5: GOSUB
     6000: RETURN
5090 M = 50:N = 75:DE = 9: GOSUB
     6000: RETURN
5095 M = 43:N = 55:DE = 11: GOSUB
     6000: RETURN
6000  REM  MAIN SUBROUTINE
6001 MI = B
6005  FOR I = 1 TO DE:V = 2 * M +
     B:W = N + 3
6007  IF M > 254 THEN A = 1
6008  IF N > 254 THEN B = 1
6010  POKE 776,V: POKE 777,W: CALL
     778
6015 A = A + 1:B = B + 1: NEXT I
6018 B = MI
6020  RETURN
6100  REM  SOUND SUBROUTINE
6200  POKE 778,173: POKE 779,48: POKE
     780,192: POKE 781,136: POKE
     782,208: POKE 783,5: POKE 78
     4,206: POKE 785,9: POKE 786,
     3: POKE 787,240: POKE 788,9:
      POKE 789,202: POKE 790,208
6300  POKE 791,245: POKE 792,174:
      POKE 793,8: POKE 794,3: POKE
     795,76: POKE 796,10: POKE 79
     7,3: POKE 798,96: POKE 799,0

6500  GOTO 10
7000  END
```

5-9 전화번호부

이 프로그램은 전화번호 관리 프로그램이다. 전화번호를 성격별로 구분해 놓았으므로 검색이 용이하고 한개씩 또는 알파벳에 따른 그룹별로 전화번호부를 인쇄할 수 있게 되어 있다.

프로그램은 메뉴식으로 사용하기 편하게 되어 있으며 수정도 손쉽게 가능하다.

프로그램을 실행시키면 화면에는 150개의 리스팅(listing)중 수록된 정보가 몇개, 남아있는 것이 몇개라고 표시한다. 또 현재 프린트 중인지 아닌지의 여부를 가린 다음 메인메뉴가 나타난다.

```
PROGRAM CURRENTLY CONTAINS 12 LISTINGS
ROOM FOR 138 ADDITIONAL LISTINGS

THE PRINTER IS  OFF

YOU MAY CHOOSE FROM THE FOLLOWING:
```

```
1  SEARCH FOR A LISTING
2  LIST THE ENTIRE FILE
3  ENTER NEW LISTINGS
4  DELETE A LISTING
5  EDIT A LISTING
6  PRINTER ON/OFF
7  EXIT THIS PROGRAM

          WHICH:
```



원하는 항목의 번호를 쳐 넣으
이것이 이 프로그램을 행할 수 있는 기능이다. 예를 들어 새로운 자료를 입력하고자 하면 3
면 해당되는 기능을 수행하게 된다. 그러면 수록될 이름을 타이핑한
번을 누르면 다음과 같은 지시사항이 나타난다.
다.

```
ENTER NEW NAME AND PRESS 'RETURN'.
(USE NO COMMAS, COLONS OR QUOTES.
TWENTY-FIVE CHARACTERS, MAXIMUM.)

-> KWACK SUNG KEUN'
```

컴퓨터가 요구하고 입력되면 확인까
그리고 다시 전화번호를 입력시켜 줄 것을 전화번호가 등록이
지 한 후, 확인이 끝나면 리스팅되어 전화번호부에 이 새로운
된다.
수정하고 싶은
이것은 전화번호의 변경이 생겼을 때의 수정도 쉽게 가능하다. 고치고 싶은 항목을 치고
자료의 코드를 입력하면 해당 내용이 화면에 표시되고
수정된 내용을 기록하면 된다.
프린트 역시 코드 또는 이름별로
기록의 삭제도 거의 같은 방법으로 가능하다.
가능하며 레코드 하나 또는 그룹별로 프린트가 가능하다.
행번호
여기서 기록자료의 DATA는 리스트가 길어지는 관계로 생략하였으니
201에서 350까지 똑같은 방식으로 150개의 데이타 저장장소를 직접 타이핑하기
바란다.

```
100  REM  ***********************
101  REM  *                     *
102  REM  *  TELEPHONE NUMBER   *
103  REM  *                     *
104  REM  ***********************
105  REM
106  REM
200  DATA 1012
201  DATA /......................
     .................
```

```
202  DATA /......................
     .................
203  DATA /......................
     .................
 :
 :
350  DATA /......................
     .................
```

```
400  REM

SET PRINTER SLOT


410  LET SL = 1
450  REM

DISK SYSTEM??


460  IF  PEEK (994) +  PEEK (1001
     ) = 192 THEN DISK = 1
470  IF DISK THEN  PRINT  CHR$ (4
     )"NOMON C,I,O": HOME
500  REM

GET ADDRESS OF FIRST DAT
     A STATEMENT


510  LET START = ( PEEK (103) +  PEEK
     (104) * 256 + 458)
550  REM

SET FAST/SLOW SEARCH




560  LET OP = 0
570  GOTO 25000
700  REM

ACTIVATE PRINTER


710  PR# SL: RETURN
750  REM

DE-ACTIVATE PRINTER


760  PR# 0: RETURN
800  REM

END ROUTINE

810  IF MF = 0 THEN 890
820  IF DISK THEN  PRINT  CHR$ (4
     )"SAVE PHONE LIST": TEXT : HOME
     : VTAB 22: CALL 976: END
830  HOME : VTAB 11: PRINT "INSER
     T A NEW TAPE INTO THE RECORD
     ER, SET";: PRINT "ON RECORD,
      AND PRESS 'RETURN' ";: INPUT
     "";S$
840  FOR I = 1 TO 1500: NEXT : SAVE
     : FOR I = 1 TO 2000: NEXT : SAVE

850  HOME : VTAB 12: PRINT "THE P
     ROGRAM AND DATA HAVE BEEN SA
     VED!!": VTAB 22: END
890  TEXT : HOME : VTAB 22: END
1000  REM

SEARCH


1005  GOSUB 3000: IF OP = 1 THEN
     1010
1006  HOME : VTAB 9: HTAB 5: PRINT
     "WOULD YOU LIKE TO SEARCH BY
     ": PRINT : HTAB 10: PRINT "1
      WORD (FAST)": HTAB 7: PRINT
     "OR 2  CHARACTER (SLOW)"
1007  VTAB 14: CALL  - 868: HTAB
     20: INPUT "WHICH: ";SK$:SK =
      VAL (SK$): IF SK <  > 1 AND
     SK <  > 2 OR  LEN (SK$) > 1 THEN
      PRINT "": GOTO 1007
1010  HOME : VTAB 11: HTAB 5: PRINT
     "PLEASE ENTER THE SEARCH KEY
     :": VTAB 13: HTAB 5: INPUT "
     ";S$: IF S$ = "" THEN 25000
1012  IF PR THEN  GOSUB 700: PRINT
     " ": PRINT " "
1015  HOME : VTAB 1: PRINT " NAME
                         PHONE
     NUMBER": PRINT "------------
     ----------------------------
     "

1017  IF PR THEN  GOSUB 750
1018  IF S = 2 THEN 2510
1020  VTAB 12: CALL  - 868: HTAB
```

```
     12: INVERSE : PRINT "<< SEAR
     CHING >>": NORMAL :VT = 2
1025  RESTORE : READ Z
1030  LET FO = 0: FOR I = 1 TO NR

1040  READ NA$: IF  LEFT$ (NA$,1)
      =  CHR$ (92) THEN 1040
1045  IF SK = 0 OR SK = 1 THEN  IF
      LEFT$ (NA$, LEN (S$)) <  >
     S$ THEN 1075
1050  FOR II = 1 TO  LEN (NA$)
1060  IF  MID$ (NA$,II, LEN (S$))
      = S$ THEN FO = FO + 1: GOSUB
     1100: GOTO 1075
1070  NEXT II
1075  NEXT I: IF FO THEN 1200
1080  VTAB 12: CALL  - 868: HTAB
     12: INVERSE : PRINT "<< NOT
     FOUND >>": NORMAL : IF PR THEN
      GOSUB 700: FOR S = 1 TO 4: PRINT
      " ": NEXT : GOSUB 750
1090  PRINT  CHR$ (7): FOR I = 1 TO
      1500: NEXT I: GOTO 25000
1099  STOP
1100  REM

GOT ONE!

1102  LET VT = VT + 1: IF VT >  =
      21 AND PR <  > 1 THEN  GOSUB
      1300
1104  IF FO = 1 THEN  VTAB 12: CALL
      - 868
1105  IF PR THEN  GOSUB 700
1110  VTAB VT: HTAB 1: PRINT NA$
1120  IF PR THEN  GOSUB 750
1140  IF I <  = NR THEN  RETURN
1200  IF PR THEN  GOSUB 700: PRINT
      " ": PRINT " ": PRINT " ": PRINT
      " ": GOSUB 750
1210  VTAB 22: CALL  - 868: PRINT
      "THAT'S ALL OF THEM! - PRESS
       'RETURN' ";: INPUT "";S$: GOTO
      25000
1299  STOP
1300  REM

GOT A FULL PAGE

1310  VTAB 22: CALL  - 958: PRINT
      "ENTER 'RTN' TO CONT - 'ESC'
      FOR MENU ";: GET T$
1320  IF T$ =  CHR$ (27) THEN  POP
     : GOTO 25000
1330  IF T$ =  CHR$ (13) THEN  VTAB
     3: HTAB 1: CALL  - 958:VT =
     3: RETURN
1340  PRINT  CHR$ (7): GOTO 1300
2499  STOP
2500  REM

LIST EVERYTHING

2504  GOSUB 3000
2505  GOTO 1012
2510  RESTORE : READ Z
2530  LET VT = 2:FO = 0: FOR I =
     1 TO NR
2540  READ NA$: IF  LEFT$ (NA$,1)
      =  CHR$ (92) AND I <  = NR THEN
     2540
2542  GOSUB 1100
2545  IF VT >  = 22 THEN VT = 2: VTAB
     3: CALL  - 958
2550  NEXT I
2560  GOTO 1200
3000  REM

IS DATA FILE EMPTY?

3010  HOME : RESTORE : IF NR = 0 THEN
      VTAB 12: HTAB 9: INVERSE : PRINT
      "<< NO NAMES ON FILE  >>": NORMAL
     : FOR I = 1 TO 2000: NEXT I:
      POP : GOTO 25000
3020  RETURN
4000  REM

ENTER LISTINGS
```

```
4010  HOME : IF NR > = 150 THEN
      VTAB 12: HTAB 12: PRINT "DA
      TA TABLE FULL!": FOR I = 1 TO
      2000: NEXT : GOTO 25000
4020  POKE 32,3: VTAB 10: CALL  -
      958: PRINT "ENTER NEW NAME A
      ND PRESS 'RETURN'.": PRINT "
      (USE NO COMMAS, COLONS OR QU
      OTES.": PRINT "TWENTY-FIVE C
      HARACTERS, MAXIMUM.)"
4030  VTAB 14: INPUT "-> ";A$: GOSUB
      6100:NN$ = B$: IF NN$ = "" AND
      S = 5 THEN  POKE 32,0: POP :
      GOTO 25000
4035  IF NN$ = "" THEN  POKE 32,0
      : GOTO 25000
4040  IF  LEN (NN$) > 25 THEN  PRINT
      "": GOTO 4020
4050  VTAB 10: CALL  - 958: PRINT
      "ENTER NEW NUMBER THEN PRESS
      'RETURN'.": PRINT "(USE THE
      FORMAT: XXX XXX XXXX)"
4060  VTAB 13: INPUT "-> ";A$: GOSUB
      6100:NP$ = B$: IF NP$ = "" AND
      S = 5 THEN  POKE 32,0: POP :
      GOTO 25000
4065  IF NP$ = "" THEN  POKE 32,0
      : GOTO 25000
4070  IF  LEN (NP$) < > 12 THEN
      PRINT "": GOTO 4050
4100  REM

IS ENTRY CORRECT?


4110  TEXT : HOME : VTAB 8: PRINT
      "HERE IS THE ";: INVERSE : PRINT
      "NAME";: NORMAL : PRINT " EN
      TRY:": VTAB 10: HTAB 14: PRINT
      NN$
4120  VTAB 12: PRINT "AND THE ";:
      INVERSE : PRINT "PHONE NUMB
      ER";: NORMAL : PRINT ":": VTAB
      14: HTAB 14: PRINT NP$
4140  VTAB 18: HTAB 8: CALL  - 86
      8: INPUT "ARE THEY CORRECT (
      Y/N) ";S$: IF  LEFT$ (S$,1) =
      "Y" THEN 4180
4150  IF  LEFT$ (S$,1) < > "Y" AND
      S = 5 THEN  POP : GOTO 6000
4160  IF  LEFT$ (S$,1) < > "N" THEN
      PRINT "": GOTO 4100
4170  GOTO 4000
4175  IF S = 5 THEN 4200
4180  IF S = 5 THEN 4200
4190  LET UD = 1
4200  REM

PAD TO 40 CHARACTERS


4210  FOR I =  LEN (NN$) TO (39 -
      LEN (NP$)):NN$ = NN$ + ".":
      NEXT
4220  LET NN$ = NN$ + NP$: IF S =
      5 THEN  RETURN
4250  REM

UPDATE DATA NUMBER


4260  RESTORE : READ T:T = T + UD
      :CU = (START - 11)
4270  LET T$ =  STR$ (T): FOR I =
      1 TO  LEN (T$)
4280  POKE (CU + I), ASC ( MID$ (
      T$,I,1)): NEXT
4290  IF S = 4 THEN  RETURN
4300  REM

FIND EMPTY DATA LINE


4310  READ T$: IF  LEFT$ (T$,1) <
      > CHR$ (92) THEN 4310
4320  LET LN =  PEEK (123) +  PEEK
      (124) * 256
4350  REM

CALCULATE OFFSET


4360  LET CU = (START + ((LN - 20
      1) * 46))
4400  REM
```

```
POKE DATA LINE


4410  FOR I = CU TO (CU + 39)
4420  POKE I, ASC ( MID$ (NN$,I +
     1 - CU,1)): NEXT
4450  REM

SET UPDATE FLAG


4460  LET MF = MF + 1: IF S = 4 OR
     S = 5 THEN  RETURN
4470  GOTO 25000
4499  STOP
5000  REM

DELETE DATA CONTENTS


5005  HOME : GOSUB 3000
5010  HOME : VTAB 10: PRINT "PLEA
     SE ENTER THE NAME OF THE PAR
     TY YOU": PRINT "WISH DELETED
      FROM THE DIRECTORY AND": PRINT
     "PRESS 'RETURN'."
5020  VTAB 14: INPUT "-> ";K$: IF
     K$ = "" AND S = 5 THEN  POP
     : GOTO 25000
5030  IF K$ = "" THEN 25000
5050  REM

FIND AN EXACT MATCH


5060  RESTORE : READ Z:UU = 1
5070  READ NA$: IF  LEFT$ (NA$, LEN
     (K$)) = K$ THEN 5100
5075  IF  LEFT$ (NA$,1) =  CHR$ (
     92) AND UU <  = NR THEN 5070

5080 UU = UU + 1
5081  IF UU <  = NR THEN 5070
5082  HOME : GOTO 25000
5100  REM

FOUND A MATCH

5110  LET LN =  PEEK (123) +  PEEK
     (124) * 256:UU = UU + 1
5120  HOME : VTAB 11: PRINT NA$: IF
     S = 5 THEN  RETURN
5130  VTAB 13: CALL  - 868: INPUT
     "DO YOU WISH TO DELETE THIS
     DATA (Y/N) ";S$: IF  LEFT$ (
     S$,1) = "Y" THEN 5200
5140  IF  LEFT$ (S$,1) <  > "N" THEN
      PRINT "": GOTO 5130
5150  GOTO 5081
5200  REM

OK THEN, KILL IT!


5210  LET UD =  - 1: GOSUB 4250
5220  LET CU = (START + ((LN - 20
     1) * 46))
5230  POKE CU, ASC ( CHR$ (92))
5240  GOSUB 4450: GOTO 25000
6000  REM

EDIT A DATA LINE


6010  HOME : GOSUB 3000
6012  HOME : VTAB 10: PRINT "PLEA
     SE ENTER THE NAME OF THE PAR
     TY WHOSE";: PRINT "LISTING Y
     OU WISH TO EDIT AND PRESS": PRINT
     "'RETURN'.": GOSUB 5020
6020  VTAB 13: CALL  - 868: INPUT
     "DO YOU WISH TO EDIT THIS DA
     TA (Y/N) ";S$: IF S$ = "" THEN
     25000
6030  IF  LEFT$ (S$,1) = "Y" THEN
     NN$ = NA$: GOTO 6050
6035  IF  LEFT$ (S$,1) = "N" THEN
     GOSUB 5070
6040  PRINT "": GOTO 6020
6050  REM

GO EDIT THE LINE


6060  HOME : VTAB 8: PRINT NN$: GOSUB
```

```
   4020: GOSUB 4360
6070  GOTO 25000
6100  REM

REMOVE CONTROL CHARS.


6105 B$ = ""
6107  IF  NOT  LEN (A$) THEN  RETURN

6110  FOR I = 1 TO  LEN (A$)
6120  IF  ASC ( MID$ (A$,I,1)) >
     26 THEN B$ = B$ +  MID$ (A$,
     I,1)
6130  NEXT I
6140  RETURN
24000  REM

PRINTER ON/OFF


24010  LET PR =  NOT PR: GOTO 250
     50
24020  STOP
25000  REM

MAIN MENU


25005  RESTORE : READ NR:NR = NR -
     1000
25010  TEXT : HOME : VTAB 1: PRINT
     "---------------------------
     -------------";: INVERSE
25020  PRINT " KWACK SUNG KEUN CO
     MPUTER TELEPHONE LIST";: NORMAL
     : PRINT "-------------------
     ---------------------"
25030  PRINT "PROGRAM CURRENTLY C
     ONTAINS "NR" LISTING";: IF N
     R <  > 1 THEN  PRINT "S";
25040  PRINT : PRINT "ROOM FOR "(
     150 - NR)" ADDITIONAL LISTIN
     GS"
25050  VTAB 8: PRINT "THE PRINTER
     IS";: IF  NOT PR THEN  PRINT
     "  OFF";
25055  IF  NOT PR THEN  VTAB 9: PRINT
     "

          ": GOTO 25070
25060  PRINT " ";: FLASH : PRINT
     " ON ": NORMAL
25070  VTAB 10: PRINT "YOU MAY CH
     OOSE FROM THE FOLLOWING:": POKE
     32,5: PRINT : PRINT
25080  PRINT "1  SEARCH FOR A LIS
     TING": PRINT "2  LIST THE EN
     TIRE FILE"
25090  PRINT "3  ENTER NEW LISTIN
     GS": PRINT "4  DELETE A LIST
     ING"
25100  PRINT "5  EDIT A LISTING":
     PRINT "6  PRINTER ON/OFF"
25110  PRINT "7  EXIT THIS PROGRA
     M": POKE 32,0: PRINT
25120  POKE 32,0: PRINT
25130  VTAB 22: HTAB 20: CALL  -
     868: INPUT "WHICH: ";S$:S =
     VAL (S$)
25140  IF S < 1 OR S > 7 THEN  PRINT
     CHR$ (7): GOTO 25130
25150  ON S GOTO 1000,2500,4000,5
     000,6000,24000,800
25160  STOP
65533  REM

END
```



5-10 전자올갠(Electronic Organ)

키보드의 CTRL-G를 치면 「삑」소리가 난다는 것은 이미 앞장에서 설명하였다. 물론 CTRL-G를 PRINT문속에 집어 넣을 수 있다. 그러나 제어문자(control character)는 프로그램 LIST에 나타나지 않는다. 이때 CHR$(7)이라는 ASCII코드로 CTRL-G와 같은 작용을 시킬 수 있다. 즉

```
10 PRINT "I CTRL-G AM CTRL-G A CTRL-G BOY. CTRL-G "
```

는

```
10 PRINT "I"; CHR$ (7);"AM"; CHR$
   (7);"A"; CHR$ (7);"BOY."; CHR$
   (7)
```

와 똑같은 작용을 한다. 이러한 「삑」이라는 소리는 컴퓨터에 내장되어 있는 특별한 연속 진동음으로 만들어진다. 따라서 애플컴퓨터는 또 다른 여러가지 음향을 낼 수 있다. 3장에서 설명된 음악 기능처럼 1초에 몇개의 진동음이 발생하느냐에 따라 음의 고저가 결정된다. 이 진동수를 변화시켜서 컴퓨터가 만들어내는 음을 변화시킨다.

그럼 키보드의 A에서 Z까지의 26문자를 사용한 음악루틴에 대한 프로그램을 생각해 보자.

행번호 18991에서 19220은 그 작용은 알지만 그 수행과정을 모르는 Black box 루틴으로 POKE를 이용한 기계어 코드(machine code)로 사용하게 된다.

또 행번호 12991에서 13670의 루틴은 자기 자신은 수행되지 못하고 다른 프로그램에 부가적으로 쓰인다. 여기서 행번호 13383~13391에서 표시하듯이 음향효과를 내기 위해서는 다음의 일곱개의 변수를 지정하여야 한다.

Entry point (시작점) : 13400

Entry variables (초기변수)

- W1 각각의 길이 : >= 0
- W2 음사이의 스텝 : > 0
- W3 시 작 음 : 0 ~ 255
- W4 한 사이클내의 음의 갯수
- W5
  - 1 = cycle down
  - -1 = cycle up
  - 0 = down & up
- W6 반복할 때 잠시 멈추는 시간
- W7 반복횟수



그리고 여러가지 음을 내기 위해 DATA문을 사용하게 되는데 이 경우 행번호 13120~13150을 변경시키면 된다.

```
13120  DATA  255,242,228,215,203,
     192,181,171
13130  DATA  161,152,143,135,127,
     120,113,107
13140  DATA  101,95,90,85,80,75,7
     1,67
13150  DATA  63,59
```

위와같이 DATA는 음의 표시가 알파벳 순으로 키보드의 문자키에 맞추었기 때문에 루틴은 26반음계를 이룬다. 그러나 이런 형태의 키 배열은 쉽게 손에 익지 않으므로 알파벳순이 아니라 키보드의 좌에서 우, 아래에서 위로 순서적으로 음이 배열되도록 완성된 프로그램에서 변경시켰다. 즉 가장 낮은 음의 키는 Z이고 가장 높은 음의 키는 P가 된다.

행번호 13692에서 13920는 음의 길이를 변화시키는 루틴이다. 음은 이 루틴에 의해 계속해서 난다. 마치 전자올갠처럼 다음 키를 누를때까지 음이 계속된다. 만일 이 루틴을 프로그램상에서 삭제하면 음의 길이는 일정해지고 한번에 한가지음 밖에 내지 못한다.

```
10  REM ************************
20  REM *                      *
30  REM *  ELECTRONIC  ORGAN   *
40  REM *                      *
50  REM ************************
60  REM
100  GOSUB 13700: REM  INITIALIZE
      ORGAN ROUTINE
200  TEXT : HOME
210  PRINT "'ELECTRONIC ORGAN' US
     ING THE KEYS A/Z"
220  PRINT
230  PRINT "PRESS RETURN TO END Y
     OUR 'TUNE'..."
240  PRINT "PRESS ESC TO STOP PLA
     YING..."
250  PRINT
300  GET Z$
310  IF Z$ =  CHR$ (13) THEN 500:
      REM  RETURN
320  IF Z$ =  CHR$ (27) THEN  END
     : REM  ESC
330  IF Z$ < "A" OR Z$ > "Z" THEN
     300: REM  IGNORE OTHER KEYS
```

```
340  PRINT Z$;
350 WP =  ASC (Z$) - 64: REM  CON
     VERT TO A/Z
360 WD = 50: REM  DURATION
370  GOSUB 13700
380  GOTO 300
500  PRINT  VTAB 18
510  PRINT "PRESS RETURN TO TRY A
     GAIN...";
520  GET Z$
530  IF Z$ =  CHR$ (27) THEN  END
     : REM  ESC
540  GOTO 200
12991 :
12992 :
12993  REM  * SOUND A PITCH FOR A
      SET DURATION *
12994  REM  ENTRY : WP PITCH #
12995  REM           (WP=0 & RO
     UTINE NOT LOADED = INITIALIZ
     ATION ONLY)
12996  REM           WD DURATION
12999 :
```

```
13000  IF WR% THEN 13200: REM  CH
     ECK IF SOUND ROUTINE ALREADY
      LOADED
13010 WR% = 800:WP% = 799:WD% = 7
     97: REM  SOUND,PITCH,DURATIO
     N ADDRESS
13020 Z = 13100: GOSUB 19000: REM
      SET READ DATA POINTER
13050 Z = WR%: REM  LOAD SOUND RO
     UTINE
13060  READ Z1: IF Z1 > = 0 THEN
      POKE Z,Z1:Z = Z + 1: GOTO 1
     3060
13070  IF WP = 0 THEN  RETURN : REM
      TRAP FOR INITIALIZATION ONL
     Y
13097 :
13098  REM  SOUND ROUTINE WRITTEN
      IN MACHINE CODE
13099 :
13100  DATA  172,31,3,185,73,3,14
     1,31,3,160,0,238,29,3,238,30
     ,3,174,31,3,173,48,192
13110  DATA  136,208,10,206,29,3,
     208,5,206,30,3,240,5,202,240
     ,234,208,238,96
13117 :
13118  REM  PITCHES
13119 :
13120  DATA  171,203,228,152,90,1
     43,135,127
13130  DATA  67,120,113,107,181,1
     92,63,59
13140  DATA  101,85,161,80,71,215
     ,95,242
13150  DATA  75,255
13190  DATA  -1: REM  FLAG TO STO
     P READING DATA
13200 Z% = WD / 256: POKE WD%,WD -
     256 * Z%: POKE WD% + 1,Z%: REM
      DURATION
13210  POKE WP%,WP: REM  PITCH #
13220  CALL WR%
13230  RETURN
13292 :
13293  REM  * PLAY STRING OF ALPH
     ABETIC LETTERS *
13294  REM  ENTRY : Z$ STRING
13295  REM          WD DURATION
13299 :
13300  IF  LEN (Z$) = 0 THEN  RETURN
      : REM  EMPTY STRING
13310  FOR W = 1 TO  LEN (Z$)
13320 WP =  ASC ( MID$ (Z$,W,1)) -
      64: REM  NEXT LETTER
13330  IF WP > = 1 AND WP < = 2
      6 THEN  GOSUB 13000: REM  PL
      AY IF IN RANGE
13340  NEXT
13350  RETURN
13382 :
13383  REM  * SOUND EFFECTS *
13384  REM  ENTRY : W1 LENGTH OF
      EACH TONE (>=0)
13385  REM           W2 STEP BETWE
      EN TONE   (> 0)
13386  REM             (W2=0 & RO
      UTINE NOT LOADED = INITIALIZ
      ATION ONLY)
13387  REM            W3 STARTIN
      G TONE    (0/255)
13388  REM            W4 # OF TO
      NES IN CYCLE
13389  REM            W5 1=CYCLE
      DOWN; -1=UP; 0=DOWN & UP
13390  REM            W6 PAUSE B
      ETWEEN REPETITIONS OF CYCLE
13391  REM            W7 # OF RE
      PETITIONS OF CYCLE
13399 :
13400  IF WE% THEN 13500: REM  CH
      ECK IF EFFECTS ALREADY LOADE
      D
13410  IF WR% = 0 THEN WP = 0: GOSUB
      13000: REM  LOAD SOUND ROUTI
      NE IF NECESSARY
13420 WE% = 809: REM  EFFECTS ADD
      RESS
13430  IF W2 = 0 THEN  RETURN : REM
      TRAP FOR INITIALIZATION ONL
      Y
13500 WH% = W1 / 256:WL% = W1 - 2
      56 * WH%: REM  DURATION AS T
      WO BYTES
13510  IF W2 < = 0 THEN W2 = 1: REM
      FORCE VALID W2
```

```
13520  IF W3 < 0 THEN W3 = 0: REM
     FORCE VALID W3
13530  FOR Z = 1 TO W7: REM  # OF
     REPETITIONS
13540 Z% = W3 + W2 * W4: IF W5 <
     0 THEN 13600: REM  TRAP FOR
     UP ONLY
13550  FOR Z1 = W3 TO Z% STEP W2:
     REM  CYCLE DOWN
13560  IF Z1 <  = 255 THEN  POKE
     WP%,Z1: POKE WD%,WL%: POKE W
     D% + 1,WH%: CALL WE%: REM  N
     EXT TONE IS IN RANGE
13570  NEXT
13600  IF W5 > 0 THEN 13650: REM
     TRAP FOR DOWN ONLY
13610  FOR Z1 = Z% TO W3 STEP  -
     W2: REM  CYCLE UP
13620  IF Z1 <  = 255 THEN  POKE
     WP%,Z1: POKE WD%,WL%: POKE W
     D% + 1,WH%: CALL WE%: REM  N
     EXT TONE IS IN RANGE
13630  NEXT
13650  FOR Z1 = 1 TO W6: NEXT : REM
     PAUSE BETWEEN CYCLE
13660  NEXT
13670  RETURN
13692 :
13693  REM  ** O R G A N **
13694  REM  ENTRY : WP PITCH #
13695  REM          (WP=0 & ROUTI
     NE NOT LOADED = INITIALIZATI
     ON ONLY)
13699 :
13700  IF WS% THEN 13900: REM  CH
     ECK IF ORGAN ROUTINE ALREADY
     LOADED
13710 WS% = 882: REM  ORGAN ADDRE
     SS
13720  IF WR% = 0 THEN W = WP:WP =
     0: GOSUB 13000:WP = W: REM
     LOAD SOUND ROUTINE (SAVING P
     ITCH)
13730 Z = 13800: GOSUB 19000: REM
     SET READ DATA POINTER
13750 Z = WS%: REM  LOAD ORGAN RO
     UTINE
13760  READ Z1: IF Z1 >  = 0 THEN
     POKE Z,Z1:Z = Z + 1: GOTO 1
     3760
13770  IF WP = 0 THEN  RETURN : REM
     TRAP FOR INITIALIZATION ONL
     Y
13797 :
13798  REM  ORGAN ROUTINE WRITTEN
     IN MACHINE CODE
13799 :
13800  DATA  172,31,3,185,73,3,14
     1,31,3,173,0,192,48,14,174,3
     1,3,173,48,192
13810  DATA  136,208,0,202,240,23
     9,208,248,96
13890  DATA  -1: REM  FLAG TO STO
     P READING DATA
13900  POKE WP%,WP: REM  PITCH #
13910  CALL WS%
13920  RETURN
18991 :
18992 :
18993  REM  * NEXT READ FROM ANY
     LINE Z *
18994  REM  ENTRY : Z LINE #
18999 :
19000  IF YR% THEN 19200: REM  CH
     ECK IF NEXT DATA ROUTINE ALR
     EADY LOADED
19010 YR% = 770: REM  NEXT DATA A
     DDRESS
19097 :
19098  REM  NEXT DATA ROUTINE WRI
     TTEN IN MACHINE CODE
19099 :
19100  POKE 770,173: POKE 771,0: POKE
     772,3: POKE 773,133: POKE 77
     4,80: POKE 775,173: POKE 776
     ,1: POKE 777,3
19110  POKE 778,133: POKE 779,81:
      POKE 780,32: POKE 781,26: POKE
     782,214: POKE 783,165: POKE
     784,155: POKE 785,24
19120  POKE 786,105: POKE 787,4: POKE
     788,133: POKE 789,125: POKE
     790,165: POKE 791,156: POKE
     792,105: POKE 793,0
19130  POKE 794,133: POKE 795,126
     : POKE 796,96
```

```
19200 Z% = Z / 256: POKE YR% - 2,
     Z - 256 * Z%: POKE YR% - 1,Z
     %: REM  LINE #
19210  CALL YR%
19220  RETURN
```



부록 A. 컴퓨터의 기능과 BASIC의 대조

기 능	금성 MIGHTY	PC6001	PC8001	PC8801
키 보 드				
ENTER	RETURN or ENTER	RETURN	RETURN	<===
BREAK	CTRL-C or STOP	STOP	STOP	STOP
BACKSPACE	DEL	DEL	INS DEL	INS DEL
ESC	ESC	ESC	ESC	ESC
HOLD	CTRL-S	ESC	ESC	CTRL-S
디스플레이				
1행의 문자수	80	32	36/40/72/80	40/80
1화면의 행수	24	16	20/25	20/25
절환 명령	WIDTH 80	–	WIDTH 80, 25	WIDTH 80, 20
BASIC				
명칭	MBASIC	N60-BASIC	N-BASIC /DISK BASIC	N88-BASIC /DISK BASIC
에디터방식	LINE	SCREEN	SCREEN	SCREEEN
프롬프트(prompt)	OK	OK	OK	OK

* LIST 때만 유효

key word	금성 MIGHTY	PC6001	PC8001	PC8801
ABS	Y	Y	Y	Y
AND	Y	Y	Y	Y
ASC	Y	Y	Y	Y
AUTO	Y	–	Y	Y
CHR$	Y	Y	Y	Y
CLEAR	Y	Y	Y	Y
CLOAD	Y	Y	Y	LOAD"CAS:"
CLS	PRINT CHR$(27); CHR$(96)	Y	CHR$(12)	Y
CONT	Y	Y	Y	Y
COS	Y	Y	Y	Y
CSAVE	Y	Y	Y	SAVE"CAS:"
DATA	Y	Y	Y	Y
DEFDBL	Y	–	Y	Y
DEFINT	Y	–	Y	Y
DEFSNG	Y	–	Y	Y
DELETE	Y	–	Y	Y
DIM	Y	Y	Y	Y
ELSE	Y	–	Y	Y
END	Y	Y	Y	Y
EXP	Y	Y	Y	Y
FIX	Y	–	Y	Y
FOR..TO..NEXT	Y	Y	Y	Y
GOSUB	Y	Y	Y	Y
GOTO	Y	Y	Y	Y
IF..THEN	Y	Y	Y	Y
INKEY$	Y	Y	Y	Y
INPUT	Y	Y	Y	Y
INT	Y	Y	Y	Y
LEFT$	Y	Y	Y	Y
LEN	Y	Y	Y	Y
LET	Y	Y	Y	Y

* Y는 기능을 갖고 있음. –는 기능이 없음을 표시

MZ-80B	PASOPIA	TRS-80 I	TRS-80 II	APPLE II
CR	RETURN	ENTER	ENTER	RETURN
SHIFT BREAK	STOP	BREAK	BREAK	CTRL - C
INST DEL	DEL	BACKSPACE	BACKSPACE	←
–	ESC	SHIFT - ↑	ESC	ESC
SPACE *	CTRL - S	SHIFT - @	HOLD	CTRL - S *
40/80	38/80	32/64	40/80	40
25	20/25	16	24	24
CONSOLE C80	WIDTH 80	CHR$(23), CLS	CHR$(30), CHR$(31)	–
SB-5520 /SB-6520	T-BASIC /DISK BASIC	L-2 BASIC /DISK BASIC	TRS-80 Model II BASIC	APPLESOFT /DISK BASIC
SCREEN	SCREEN	LINE	LINE	LINE
Ready_	OK_	READY>_	READY>_	READY_

*LIST때만 유효

MZ-80B	PASOPIA	TRS-80 I	TRS-80 II	*APPLE II
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y
Y	Y	Y	Y	–
Y	Y	Y	Y	Y
–	Y	Y	Y	Y
CHR$(6)	Y	Y	–	LOAD
Y	Y	Y	Y	HOME
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y
Y	Y	Y	–	SAVE
–	Y	Y	Y	Y
–	Y	Y	Y	–
–	Y	Y	Y	–
–	Y	Y	Y	–
Y	Y	Y	Y	DEL
–	Y	Y	Y	Y
Y	Y	Y	Y	–
Y	Y	Y	Y	Y
–	Y	Y	Y	Y
Y	Y	Y	Y	–
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y
GET	Y	Y	Y	Y
Y	Y	Y	Y	GET
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y
Y	Y	Y	Y	Y

부록 B. 컴퓨터와 BASIC 비교표

대표적 키워드	금성 MIGHTY	PC6001	PC8001
LIST	Y	Y	Y
LLIST	Y	Y	Y
LOAD	Y	–	Y
LOG	Y	Y	Y
LPRINT	Y	Y	Y
MID$	Y	Y	Y
NEW	Y	Y	Y
NOT	Y	Y	Y
ON GOTO, GOSUB	Y	Y	Y
OR	Y	Y	Y
PRINT	Y	Y	Y
PRINT@	PRINT CHR $ (27), CHR $ (102), CHR $ (31+X), CHR $ (31+Y)	LOCATE	LOCATE
READ	Y	Y	Y
REM(')	Y	Y	Y
RESTORE	Y	Y	Y
RETURN	Y	Y	Y
RIGHT $	Y	Y	Y
RND	Y	Y	Y
RUN	Y	Y	Y
SAVE	Y	–	Y
SGN	Y	Y	Y
SIN	Y	Y	Y
SQR	Y	Y	Y
STEP	Y	Y	Y
STOP	Y	Y	Y
STRING $	Y	–	Y
STR $	Y	Y	Y
TAB	Y	Y	Y
TAN	Y	Y	Y
TIME $	–	TIME	Y
USING	Y	–	Y
VAL	Y	Y	Y
(+) PLUS	Y	Y	Y
(−) MINUS	Y	Y	Y
(*) MULTIPLY	Y	Y	Y
(/) DIVIDE	Y	Y	Y
(↑) POWER	Y	Y	Y
CHR $ 의 機能			
CR/LF	CHR $ (13) CHR $ (10)	–	–
커서·온	–	–	LOCATE x, y, 1
커서·업 (cursor off)	–	–	LOCATE x, y, 0
백스페이스	CHR $ (8)	CHR $ (7)	–
커서를 한字 왼쪽으로	CHR $ (27) CHR $ (68)	CHR $ (4)	CHR $ (29)
커서를 한字 오른쪽으로	CHR $ (27) CHR $ (67)	CHR $ (3)	CHR $ (28)
커서를 한字 아래로	CHR $ (27) CHR $ (66)	CHR $ (1)	CHR $ (31)
커서를 한字 위로	CHR $ (27) CHR $ (65)	CHR $ (2)	CHR $ (30)
커서를 화면 왼쪽 위로	CHR $ (27) CHR $ (72)	CHR $ (5)	CHR $ (11)
커서를 行의 처음으로	CHR $ (13)	–	CHR $ (13)
行의 끝까지 지운다.	CHR $ (27) CHR $ (91) CHR $ (75)	–	–
화면의 끝까지 지운다	CHR $ (27) CHR $ (96)	–	–
다북레다	CHR $ (9)	–	CHR $ (9)

Y는 기능을 갖고 있음. –는 기능이 없음을 표시함.

PC8801	MZ-80B	PASOPIA	TRS-80 I	TRS-80 II	APPLE II
Y	Y	Y	Y	Y	Y
Y	LIST/P	Y	Y	Y	PR# LIST
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	PRINT/P	Y	Y	Y	PR# PRINT
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
LOCATE	CURSOR	LOCATE	Y	Y	VTAB:HTAB
Y	Y	Y	Y	Y	Y
Y	REM	Y	Y	Y	REM
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	–
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	–
Y	TI$	Y	Y	Y	–
Y	–	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y
–	–	–	CHR $ (10)	–	CHR $ (13)
LOCATE x, y, 1	–	LOCATE x, y, 1	CHR $ (14)	CHR $ (1)	–
LOCATE x, y, 0	–	LOCATE x, y, 0	CHR $ (15)	CHR $ (2)	–
CHR $ (8)	CHR $ (7)	–	–	CHR $ (8)	CHR $ (8)
CHR $ (29)	CHR $ (4)	CHR $ (29)	CHR $ (24)	CHR $ (28)	–
CHR $ (28)	CHR $ (3)	CHR $ (28)	CHR $ (25)	CHR $ (29)	–
CHR $ (31)	CHR $ (1)	CHR $ (31)	CHR $ (26)	CHR $ (10)	CHR $ (10)
CHR $ (30)	CHR $ (2)	CHR $ (30)	CHR $ (27)	CHR $ (11)	–
CHR $ (11)	CHR $ (5)	CHR $ (11)	CHR $ (28)	CHR $ (20)	–
CHR $ (13)	–	CHR $ (13)	CHR $ (29)	CHR $ (13)	–
–	–	–	CHR $ (30)	CHR $ (23)	–
–	–	–	CHR $ (31)	CHR $ (24)	–
CHR $ (9)	–	CHR $ (9)	–	CHR $ (9)	–

부록 C. 중 간 언 어

16진수	10진수	중 간 언 어	16진수	10진수	중 간 언 어	16진수	10진수	중 간 언 어
80	128	END	A4	164	LOMEM:	C8	200	+
81	129	FOR	A5	165	ONERR	C9	201	−
82	130	NEXT	A6	166	RESUME	CA	202	*
83	131	DATA	A7	167	RECALL	CB	203	/
84	132	INPUT	A8	168	STORE	CC	204	^
85	133	DEL	A9	169	SPEED=	CD	205	AND
86	134	DIM	AA	170	LET	CE	206	OR
87	135	READ	AB	171	GOTO	CF	207	>
88	136	GR	AC	172	RUN	D0	208	=
89	137	TEXT	AD	173	IF	D1	209	<
8A	138	PR#	AE	174	RESTORE	D2	210	SGN
8B	139	IN#	AF	175	&	D3	211	INT
8C	140	CALL	B0	176	GOSUB	D4	212	ABS
8D	141	PLOT	B1	177	RETURN	D5	213	USR
8E	142	HLIN	B2	178	REM	D6	214	FRE
8F	143	VLIN	B3	179	STOP	D7	215	SCRN(
90	144	HGR2	B4	180	ON	D8	216	PDL
91	145	HGR	B5	181	WAIT	D9	217	POS
92	146	HCOLOR=	B6	182	LOAD	DA	218	SQR
93	147	HPLOT	B7	183	SAVE	DB	219	RND
94	148	DRAW	B8	184	DEF	DC	220	LOG
95	149	XDRAW	B9	185	POKE	DD	221	EXP
96	150	HTAB	BA	186	PRINT	DE	222	COS
97	151	HOME	BB	187	CONT	DF	223	SIN
98	152	ROT=	BC	188	LIST	E0	224	TAN
99	153	SCALE=	BD	189	CLEAR	E1	225	ATN
9A	154	SHLOAD	BE	190	GET	E2	226	PEEK
9B	155	TRACE	BF	191	NEW	E3	227	LEN
9C	156	NOTRACE	C0	192	TAB(	E4	228	STR$
9D	157	NORMAL	C1	193	TO	E5	229	VAL
9E	158	INVERSE	C2	194	FN	E6	230	ASC
9F	159	FLASH	C3	195	SPC(	E7	231	CHR$
A0	160	COLOR=	C4	196	THEN	E8	232	LEFT$
A1	161	POP	C5	197	AT	E9	233	RIGHT$
A2	162	VTAB	C6	198	NOT	EA	234	MID$
A3	163	HIMEM:	C7	199	STEP			

부록 D. 16진수와 10진수 변환표 (Hexadecimal Conversion Table)

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	135	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

5		4		3		2		1		0	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15

부록 E. APPLE Ⅱ의 공통사양 (Common Specification)

1) 마이크로 프로세서 (micro processor)

CPU 6502
번지 지정 방식 13
clock주파수 1.023MHz
6 K bit 다이나믹 램(dynamic RAM)

2) 메모리 (memory, 記憶裝置)

최대 48K byte RAM(64K ····· language card 사용시)
12K byte ROM

3) 비디오 디스플레이 (video display)

text mode 40자×24자, 960문자(1 page)
1 문자당 5 × 7 점 매트릭스(dot matrix)
inverse flashing 가능
영문 대문자 26, 수자10, 기호28
2 page절환 가능
윈도우 세트(window set) 가능
* 低解詳度 그래픽(low resolution graphics color graphics)
40행×48행 또는 40행×(40행+text4 행)
16색
2 page 절환 가능(1 page 약 1K byte)
* 高解詳度 그래픽(high resolution graphics)
280H×192V(dot수) 또는 280H×160V×text 4 행
6 색(청, 등, 녹, 자, 백, 흑)
2 page 절환 가능(1 page 약 8K byte)

4) 입력 / 출력(I/O)

* 키보드(Keyboard)
52개 key ASCII 대문자(大文字)
code수 91
7 bit data+스트로우브
* 카세트 접속기(cassette interface, minijack)
1500 보우(baud)
입력 전압 1 Vp-p(peak to peak value)

입력 임피이던스(input impedance) 12kΩ

출력 전압 25mV (100Ω 부하시)

* 스피커(speaker, connecter Molex KK100 2P)

구경 5 cm

출력 0.5W (8 Ω)

* 게임 입력/출력(game I/O, 16P IC socket)

TTL(transistor transistor logic) 입력 3

TTL 출력 4

패들(paddle) 입력 4

스트로우브(strobo) 1

* 비디오(video) 출력

RCA pin jack

NTSC(national television system committee, 텔레비전走査方式)

composite video 신호(정극성)

흑 level 0.75

백 level 2V

출력 전압 가변(0～1 V_{p-p})

* 주변장치 스롯(peripheral slot, 50P card edge connecter × 8)

번지 버스 (address bus)

데이타 버스(data bus)

slot 0은 언어전용

5) APPLE Ⅱ의 크기(size)

387W × 457D × 133H (mm)

6) 전원 규격

입력 전압100V AC

출력전 압 + 5 V, 2.5A

− 5 V, 250mA

+ 12V, 1.5A

− 12V, 250mA

소비 전력 최대 79W

부록 F. 퍼스널 컴퓨터 공급현황 및 제품의 주요특성

공 급 처	제 품 명	제 품 특 성			가 격	전 화 문 의
		CPU	RAM	ROM	(단위 : 원)	
금 성 사	FC-30	Z-80A	2KB	8KB	129,800	583-8081~4
	FC-100	〃	18KB	28KB	399,000	
	FC-150	〃	20KB	8KB	266,200	
동 양 나 이 론	PC-8000	Z-80A 6502	64KB	24KB	407,000	771-13 (교환 : 572~5)
삼 보 컴 퓨 터	TRIGEM-20XT	6502	48KB	16KB	340,000	783-9981~6
삼보엔지니어링	SE-6001	6502	192KB	12KB	600,000	265-9385
	SE-6002	〃	64KB	24KB	450,000	
	SE-6003	〃	128KB	12KB	700,000	
삼 성 전 자	SPC-500	Z-80A	32KB	16KB	238,000	771-78
	SPC-1000	〃	70KB	32KB	469,000	(교환 : 472~4)
스포트라이트	SPOTLIGHT Ⅱ	Z-80A	48KB	16KB	363,000	713-4914~6
	SPOTLIGHT Ⅲ	〃	64KB	20KB	698,500	
에 이 스 컴 퓨 터	ACE-100	6502	64KB	12KB	300,000	792-8802
	ACE-200	〃, Z-80	64KB	12KB	960,000	792-9821
한 국 마 이 컴	MICOM Ⅱ K-plus	6502	48KB	24KB	300,000	592-1441
이 행 전 기	CYBORG-2	6502	64KB	24KB	308,000	392-6611~5
	CYBORG-3	〃, Z-80	64KB	24KB	396,000	
대 우 전 자	DPC-100	Z-80A	32KB	40KB	257,000	754-5637, 5638
	DPC-180	Z-80A	80KB	40KB	274,000	
	DPC-200	Z-80A	80KB	40KB	295,000	
C S 컴 퓨 터	STONE Ⅱ	6502	64KB	12KB	250,000	462-9985~7
	STONE-8001	Z-80	32KB	24KB	400,000	
한국맥스전자	MAXTAR Ⅱ	6502	64KB	12KB	295,900	567-1664
	MAXTAR Ⅲ	6502	64KB	12KB	371,800	

(1985년 2월 20일 현재)

공 급 처	제 품 명	제 품 특 성			가 격 (단위 : 원)	전 화 문 의
		CUP	RAM	ROM		
경성반도체(주)	BASIC－A	6502	64KB	12KB	280, 000	883－1072
고 려 시 스 템	KSYS－800	Z －80A	20KB	8KB	198, 000	782－4411～5
매 직 소 프 트	MAGIC－ Ⅱ	6502	48KB	12KB	286, 000	546－2303, 2307
정 원	HART40	6502 Z －80A	64KB	16KB	495, 000	543－2351
로 얄 컴 퓨 터	ROYAL Ⅱ	6502	48KB	12KB	200, 000	
	ROYAL DLX	〃	64KB	12KB	250, 000	
	ROYAL MAX V-64	〃	64KB	24KB	380, 000	
썬트로닉스컴퓨터	MALUM Ⅱ	6502	48KB	12KB	250,000	461－1524～5
	MALUM Ⅱ ET	〃	64KB	12KB	235,000	
	MALUM Ⅱ DLX	〃	64KB	24KB	330, 000	
흥 익 전 자	BEAR Ⅱ	6502	48KB	12KB	280,000	272－1203～4
	BEAR Ⅱe	〃	64KB	24KB	330,000	
희 망 전 자	PINEAPPLE Ⅱ	6502	64KB	20KB	170,000	272－7224
	MZ－80 C/K Ⅱ	Z－80	49KB	6 KB	220,000	
샤 프 전 자	MZ－731	Z－80A	64KB	12KB	550,000	724－1660,1629
	MZ－721	Z－80A	64KB	12KB	399,300	

※ 제품의 특성과 용도에 따라 가격면에 차이가 있으므로 상세한 것은 전화 문의를 통해 확인하시기 바랍니다. 〈가격표시는 소비자 가격을 나타냄〉 (무순)

찾 아 보 기

APPLE Ⅱ

BASIC 프로그래밍 값 5, 500 원

1984年 4月 5日 초판발행
1985年 3月 10日 재판발행

著 者 郭 成 根

發行人 劉 光 鍾

發行所 韓 國 理 工 學 社

서울特別市 西大門區 玉川洞 18 番地
전 화 : 363 - 7 6 9 7 · 6 5 9 5
대체계좌 : No . 010033 - 31 - 615070
등 록 : 1977 년 2 월 1 일 No . 9 - 92